

CONSENSUS ALGORITHM IN
SMART GRID AND COMMUNICATION NETWORKS

by

HUSAIN ABDULAZIZ ALFAGEE

B.S.E.E, Higher Institute of Engineering, 1998

A thesis submitted to the
Faculty of Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Master of Science
Electrical Engineering

2013

This thesis for the Master of Science degree by
Husain Abdulaziz Alfagee
has been approved for the Electrical Engineering Program
by

Miloje Radenkovic, Chair

Mark Golkowski

Sam Welch

June 20, 2013

Alfagee, Husain Abdulaziz (M.S., Electrical Engineering)

Consensus Algorithm in Smart Grid and Communication Networks.

Thesis directed by Professor Miloje Radenkovic

ABSTRACT

On a daily basis, consensus theory attracts more and more researches from different areas of interest, to apply its techniques to solve technical problems in a way that is faster, more reliable, and even more precise than ever before. A power system network is one of those fields that consensus theory employs extensively. The use of the consensus algorithm to solve the Economic Dispatch and Load Restoration Problems is a good example. Instead of a conventional central controller, some researchers have explored an algorithm to solve the above mentioned problems, in a distribution manner, using the consensus algorithm, which is based on calculation methods, i.e., non estimation methods, for updating the information consensus matrix.

Starting from this point of solving these types of problems mentioned, specifically, in a distribution fashion, using the consensus algorithm, we have implemented a new *advanced consensus algorithm*. It is based on the adaptive estimation techniques, such as the Gradient Algorithm and the Recursive Least Square Algorithm, to solve the same problems. This advanced work was tested on different case studies that had formerly been explored, as seen in references 5, 7, and 18. Three and five generators, or agents, with different topologies, correspond to the Economic Dispatch Problem and the IEEE 16-Bus power system corresponds to the Load Restoration Problem.

In all the cases we have studied, the results met our expectations with extreme accuracy, and completely matched the results of the previous researchers. There is little question that this research proves the capability and dependability of using the consensus algorithm, based on the estimation methods as the Gradient Algorithm and the Recursive Least Square Algorithm to solve such power problems.

The form and content of this abstract are approved. I recommend its publication.

Approved: Miloje Radenkovic

DEDICATION

I lovingly dedicate this thesis to my parents, who gave me an appreciation of learning and taught me the value of perseverance and resolve. I also dedicate this thesis to my wife for her unfaltering support and understanding while I was doing my Master's.

ACKNOWLEDGEMENT

My special thanks to my advisor, Prof. Miloje Radenkovic, for his contribution and support to my research. I also wish to thank all the members of my committee for their valuable participation and insights.

CONTENTS

Figures	x
Tables	xii
<u>Chapter</u>	
1. Introduction	1
1.1 Thesis Objectives	4
1.2 Thesis Organization	5
2. Graph and Consensus Theory	7
2.1 Graph Theory	7
2.1.1 Graph Definition	9
2.1.2 Directed and Undirected Graph	10
2.1.3 Connected and Disconnected Graph	11
2.1.4 Incidence and Adjacency Matrices	11
2.2 Consensus Theory	13
2.2.1 Consensus Definition	13
2.2.2 Consensus in Networks	13
2.3 Consensus Algorithm Applications	16
3. Economic Dispatch and Load Restoration Problems	18
3.1 Economic Dispatch Problem	18
3.2 Load Restoration Problem	20
3.3 Estimation Information Matrix in the Consensus Algorithm	22
3.4.1 Gradient Algorithm Estimation Form	23
3.4.2 Recursive Least Square Algorithm Estimation Form	24
4. Study Cases and Results	25

4.1	Three Generators System for Economic Dispatch Problem	25
4.1.1	Three Generators System with New Estimation Method using Gradient Algorithm without Power Constraints	26
4.1.2	Three Generators System with New Estimation Method using Gradient Algorithm with Power Constraints	27
4.2	Five Generators System for Economic Dispatch Problem	28
4.2.1	Five Generators System with New Estimation Method using Gradient Algorithm	29
4.2.1.1	Five Generators using GA; Guessing Initial Values $d_{ij} = d_{ji} = 0.0$	30
4.2.1.2	Five Generators using GA; Guessing Initial Values $d_{ij} = d_{ji} \neq 0.0$	31
4.2.1.3	Five Generators using GA; Guessing Initial Values $d_{ij} \neq d_{ji} \neq 0.0$	32
4.2.2	Five Generators System with New Estimation Method using Recursive Least Square Algorithm	33
4.2.2.1	Five Generators using RLS; Guessing Initial Values $d_{ij} = d_{ji} = 0.0$	34
4.2.2.2	Five Generators using RLS; Guessing Initial Values $d_{ij} = d_{ji} \neq 0.0$	35
4.2.2.3	Five Generators using RLS; Guessing Initial Values $d_{ij} \neq d_{ji} \neq 0.0$	36
4.2.3	Five Generators System with New Estimation Method using GA and RLS where the System Topology Changes with Time	37
4.2.4	Five Generators System with New Estimation Method using GA and Two Agents 4 and 5 are Become Disconnected	39
4.2.5	Five Generators System with New Estimation Method using RLS and Check the Necessity of Reaching Consensus	42
4.3	IEEE 16-Bus Test Power System for Load Restoration Problem	45
4.3.1	IEEE 16-Bus Test Power System with New Estimation Method using GA and 0-1 Knapsack Value Based	50
4.3.2	IEEE 16-Bus Test Power System with New Estimation Method using GA and 0-1 Knapsack Density Based	52
5.	Conclusion and Future Work	54

5.1 Conclusion	54
5.2 Future Work	55
<u>References</u>	56
<u>Appendix</u>	
A.1 Derivation of Recursive Least Square Algorithm Estimation (RLS)	60
A.2 Derivation of Gradient Algorithm Estimation (GA)	61
A.3 Basic Flowchart for the Estimation Algorithms (RLS and GA)	62

FIGURES

Figure

1.1 (a) Central control, and (b) distributed control connections	3
2.1 Bridges of Konigsberg City (1736)	7
2.2 The Four regions and seven bridges in Konigsberg City	8
2.3 Eulerian Circuit	8
2.4 Three generators (agents) system	9
2.5 Directed graph	10
2.6 (a) Connected graph, and (b) disconnected graph	11
2.7 (a) Adjacent vertices, and (b) adjacent edges	11
4.1 Three generators using GA; IC's and power (no power constraints)	26
4.2 Three generators using GA; IC's and power (with power constraints)	27
4.3 Five generators; different system topologies	29
4.4 Five generators using GA; IC's and power ($d_{ij} = d_{ji} = 0.0$)	30
4.5 Five generators using GA; IC's and power ($d_{ij} = d_{ji} \neq 0.0$)	31
4.6 Five generators using GA; IC's and power ($d_{ij} \neq d_{ji} \neq 0.0$)	32
4.7 Five generators using RLS; IC's and power ($d_{ij} = d_{ji} = 0.0$)	34
4.8 Five generators using RLS; IC's and power ($d_{ij} = d_{ji} \neq 0.0$)	35
4.9 Five generators using RLS; IC's and power ($d_{ij} \neq d_{ji} \neq 0.0$)	36
4.10 Five generators; changing sequence of system topology	38
4.11 Five generators using GA; IC's and power (topology changes with time)	38
4.12 Five generators using RLS; IC's and power (topology changes with time)	39
4.13 Five generators; agents 4 and 5 become disconnected	40
4.14 Five generators using GA; IC's and power (two agents 4 and 5 become disconnected)	41

4.15	Five generators; changing sequence of system topology	42
4.16	Five generators using RLS; IC's and power ($P_o = 0.01$)	43
4.17	Five generators; adjacency rows and columns' summation ($P_o = 0.01$)	43
4.18	Five generators using RLS; IC's and power ($P_o = 10$)	44
4.19	Five generators; adjacency rows and columns' summation ($P_o = 10$)	44
4.20	The 16-bus test system (source ref. [18])	45
4.21	The 16-bus system; post fault configuration (source ref. [18])	47
4.22	The 16-bus system; after fault configuration (source ref. [18])	49
4.23	The 16-bus system; information matrix-M convergence	50
4.24	The 16-bus system; total net power convergence (knapsack value based)	50
4.25	The 16-bus system; information matrix-M convergence	52
4.26	The 16-bus system; total net power convergence (knapsack density based)	52
A.1	Flowchart of solving EDP based on new estimation algorithms	62

TABLES

Table

4.1	Three generators; system parameters	25
4.2	Five generators; system parameters	28
4.3	16-Bus power system; system data	46
4.4	16-Bus power system; final converged of information matrix-M	48

1. Introduction

For more than half of a century, the existing US power grid has been providing its electric power to a wide range of customers. In fact, this system faced substantial magnitudes of challenges that were not taken into account at the time the design was originated. These challenges can be summarized in accordance with the following:

- lack of supporting the distributed generation.
- renewable energy sources which are being added to the power grid.
- lack of flexibility and adaptability.
- natural disasters.
- human operating errors.

For all of the above reasons, and to assure that the power grid will continue to provide the demanded power to its consumers as needed, the power grid should be efficient and smart enough to deal with all demeanors of ongoing and varying changes of the system topology without losing its stability, availability, et cetera.

In order to make a power system network effective and smart enough, it should implement a communication network and a control network that integrate and complement the future power system, where the system is called the *Large-Scale Network Control System* (LSNCS) [4][5]. There is no doubt that the LSNCS will be a necessity for the next series of power networks. This refers to a specific means and existence of a large number of subsystems. In this bulk system, the conventional control schematic will face severe challenges that will inevitably affect its controllability. These challenges can be summarized in the following factors [5]:

- requirement of a high level of connectivity to achieve the optimal operation

condition.

- sensitivity to failures and modeling errors.
- affect of a variety of the configurations, such as Plug-and-Play of the power grid topology which will make the smart grid topology unknown or undefined.

To overcome the drawbacks of conventional control, distribution control is more convenient for solving the identified problems. The main task for using distribution control algorithms is to make sure all system agents, or nodes, reach what is called the *consensus*. Several areas, where the consensus algorithm is applied, can be used in the improvement of the power system. For example, the consensus algorithm of a multi-vehicle system, where a vehicle is able to communicate with other vehicles upon a communication network system, is a good example [2]. Therefore, to control this LSNCS smart grid, a very strong algorithm has to be implemented in order to work correctly in the existence of a lack of or unreliable communication capability; this applies even with loss of the central control topology. Such an algorithm has to be tested in areas that resemble LSNCS, as would the Economic Dispatch Problem and the Load Restoration Problem.

To illustrate the idea of the central and distribution control, let us consider the “five buses power system”, where each bus has its load and generated power as it is shown in Figure 1.1 on the next page.

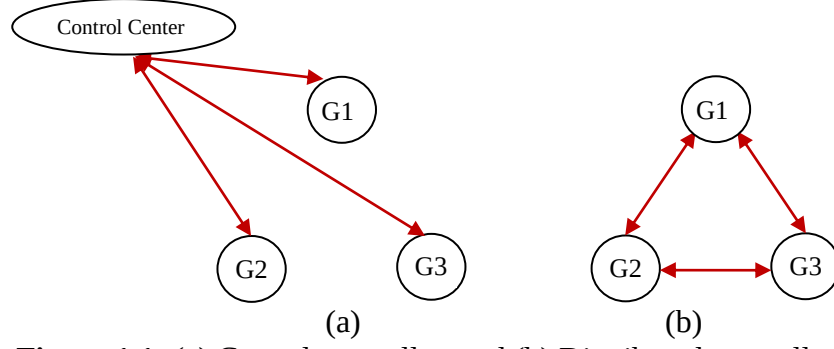


Figure 1.1: (a) Central controller, and (b) Distributed controller

The communication topology could be a conventional central controller, as shown in Figure 1.1(a), or a distributed controller as in Figure 1.1(b). In Figure 1.1(b), generator one has been selected as the consensus leader. The local controller, in each bus, will keep reporting its status to the leader. Based on the collected information, the leader will decide whether to increase or decrease the group of the consensus variable, according to the situation of a required condition [5]. The entire process mentioned is manipulated by the consensus algorithm.

The consensus algorithm is used to solve some fundamental problems of the power system, as the Economic Dispatch problem and the Load Restoration problem. Instead of the conventional central controller, Some researchers had explored consensus algorithms to solve the mentioned problems in a distribution manner [5][8][18]. The main reason of using the distribution fashion is to overcome the drawbacks of the central controller.

Starting from the point of solving the economic dispatch, and the load restoration problems in a distribution fashion, by using the consensus algorithm, we have implemented a new advanced consensus algorithm based on the adaptive estimation techniques of the Gradient Algorithm and the Recursive Least Square Algorithm to solve the same problems. This advanced work is tested on different case studies that are

proposed in the references [5][7][18] to ascertain the sets of comparatively-sourced results do coincide and are depicted accurately. These case studies, and their results, are presented in details in the Chapter 4. No doubt, this research proves the capability and dependability of using the consensus algorithm, based on the estimation methods, citing specifically, the Gradient Algorithm and the Recursive Least Square Algorithm to solve such power problems.

Briefly, our research objectives outlining this thesis are summarized in the following section:

1.1 Thesis Objectives

To fulfill the goals of this research, a set of objectives was specified inclusive in this work. These objectives are summarized in the following points:

1. Implementing a Matlab code to solve the Economic Dispatch Problem for the previous case studies that were focused on in the reference [5] without the power constraints and fulfilling the same results.
2. Implementing a Matlab code to solve the Economic Dispatch Problem and taking into account the power constraints in order to fulfill the power limitations, i.e., a condition that was mentioned in Reference [7].
3. Implementing a new Matlab code to solve the Economic Dispatch Problem in the same case studies addressing objectives 1 and 2; but this time by applying a new estimation method using the ***Gradient Algorithm*** and fulfilling the same results in objectives 1 and 2.
4. Implementing a new Matlab code to solve the Economic Dispatch Problem in the same case studies in objectives 1 and 2; but this time by applying a new

estimation method using the ***Recursive Least Square Algorithm*** and fulfilling the same results in objectives 1 and 2.

5. Repeating objectives 3 and 4 to study the case where the system topology changes with time.
6. Implementing a Matlab code to solve the Economic Dispatch Problem to study the case if some agents are found to be disconnected.
7. Implementing a Matlab code to solve the Load Restoration Problem for the case study that was presented in reference [18] and fulfilling the same results.
8. Implementing a new Matlab code to solve the Load Restoration Problem in the same case in objective 7; but this time by applying the new estimation method using the ***Gradient Algorithm*** and obtaining the same results in objective 7.

1.2 Thesis Organization

The remainder of this thesis is organized and described in methodologies as follows:

- Chapter 2: provides a very important background about the graph and consensus theories in the networks; the consensus mathematical model is presented in this chapter
- Chapter 3: presents one of the very useful applications of the consensus theory, which is solving the economic dispatch and the load restoration problems in a distributed manner; the mathematical model of the new consensus estimation method is presented in this chapter, as well
- Chapter 4: presents the study cases, and their respective results, of the estimation methods to solve the economic dispatch and the load restoration problems; also, it

contains notes and comments in direct reference and concern to the results

- Chapter 5: thesis conclusions and recommendations for future studies are presented in this chapter, which is the last chapter of the thesis

In the following chapter, the thesis starts with an introduction about the graph and the consensus theory.

2. Graph and Consensus Theory

Before going in the details of the consensus theory, there is an important concept which must be clearly understandable. This concept is called the **Graph**.

2.1 Graph Theory

If we look around us in the real world, there are many situations that can be easily described or converted to a diagram which consists of a set of points, called vertices, together with lines, called edges, which connect or join certain pairs of the points within the diagrams. For instant, the subway system, where the points could represent the stop stations, and the edges represent the rails that connect these stations. The mathematical concept of a situation of such a case provides the idea of what is called **a Graph**.

The principle of the **Graph** was born from the idea of the Königsberg Bridge Problem. The Pregel River in Prussia divided the city of Königsberg into four separate regions which were connected by seven bridges as shown in Figure 2.1 below.



Figure 2.1: Bridges of Königsberg city (1736)

A simple diagram of the bridges in the city is illustrated in Figure 2.2 below.

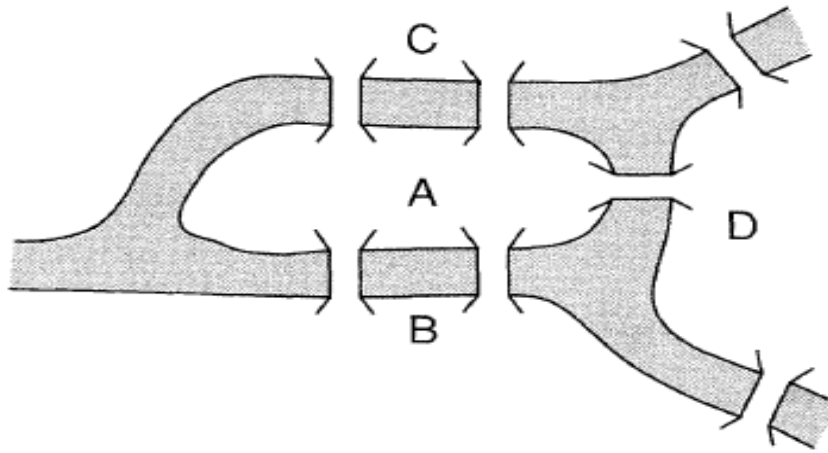


Figure 2.2: The Four regions and seven bridges in the Königsberg city

The inhabitants of the city wondered if it were possible to leave home, cross each of the seven bridges only once, and return home. The Leonhard Euler, a Swiss mathematician, (1707-1783), took the puzzle and tried to apply a mathematical method to solve the problem. He redrew each region as a vertex and each bridge as an edge connected vertices corresponding to the regions, as shown in Figure 2.3.

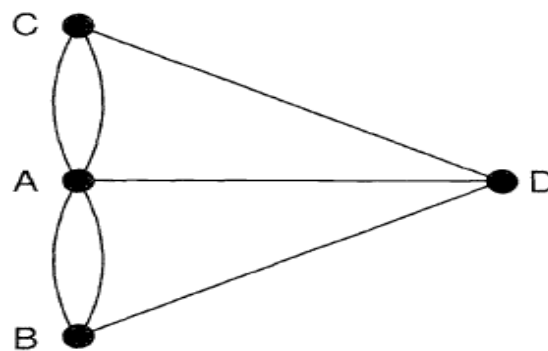


Figure 2.3: Eulerian Circuit

Figure 2.3 shows the Graph that encodes all necessary information from Figure 2.2. It is called the Eulerian circuit. Many consider the Euler's method to be the birth of the Graph theory [1][9][11].

2.1.1 Graph Definition

The **Graph** G is defined as a structure which consists a vertex set $V(G)$, and an edge set $E(G)$, and a relation that associates with each edge whose two vertices are called the endpoints. This graph is used to model the system topology of the network. The graph $G = (V, E)$ where $V = \{1, 2, \dots, n\}$ is called the vertices or the nodes. $E \subseteq V \times V$ is called the edges which is a set of unordered pairs of distinct vertices. When the two vertices $1, 2$ in $V(G)$ are endpoints of an edge, both 1 and 2 are called adjacent [1].

In the real world, **Graph** G can be an electric circuit where vertices represent sources, diodes, capacitors, et cetera, and edges represent the wires that connect them. It can be a computer network where vertices represent computers, and edges represent the communication network that connects them together. It can be the World Wide Web, where vertices represent the WebPages, and edges represent the hyperlinks, and so on. In this research, the graph is an electric power grid where the vertices represent the power generators or loads, and the edges represent the communication network that connects them as illustrated in Figure 2.4 below, specifically, for the three generators case study. It is an unordered, an undirected, and un-weighted graph.

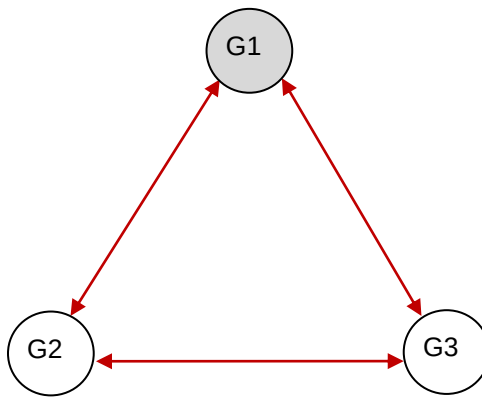


Figure 2.4: Three generators (agents) system

2.1.2 Directed and Undirected Graph

As was mentioned, Graph G is a mathematical structure consists of a set of vertices $V(G)$, and a set of edges $E(G)$, which connect the vertices according to an associated relationship. This mathematical structure can be a directed or an undirected graph. In the undirected graph, the edges that connect the vertices are not directionally fixed. The direction from a node n_i to a node n_j is represented by a line, with or without, arrowhead in both line ends, as seen in Figure 2.4 on the previous page. On the other hand, in the directed graph, also referred to as the digraph, the edge that connects two vertices is directed, or ordered, to gravitate in a specific direction from a node n_i , or the tail node, to a node n_j , called the head node. The edge direction can be easily recognized by the arrowhead that always points to the head node as illustrated in Figure 2.5 below.

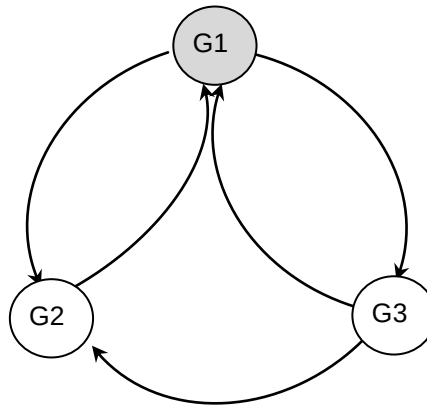


Figure 2.5: Directed graph

Formally; $G = (V, E)$ where $V = \{1, 2, \dots, n\}$ and $E \subseteq V \times V$.

Assuming that u and w are end points then:

$G = (V, E)$ will be $\begin{cases} \text{undirected} & \text{if for all } u, w \in V, \text{ i.e. } (u, w) \in E \Leftrightarrow (w, u) \in E \\ \text{directed} & \text{otherwise} \end{cases}$

2.1.3 A Connected and Disconnected Graph

The Graph G is a connected graph if, between every two vertices in the $V(G)$, there is a path linking them, otherwise the *Graph G* is called a disconnected graph. In other words, the graph is disconnected if its vertex set can be divided into two non-empty subsets where no edge connects them [10]. Figure 2.6 below illustrates the two types.

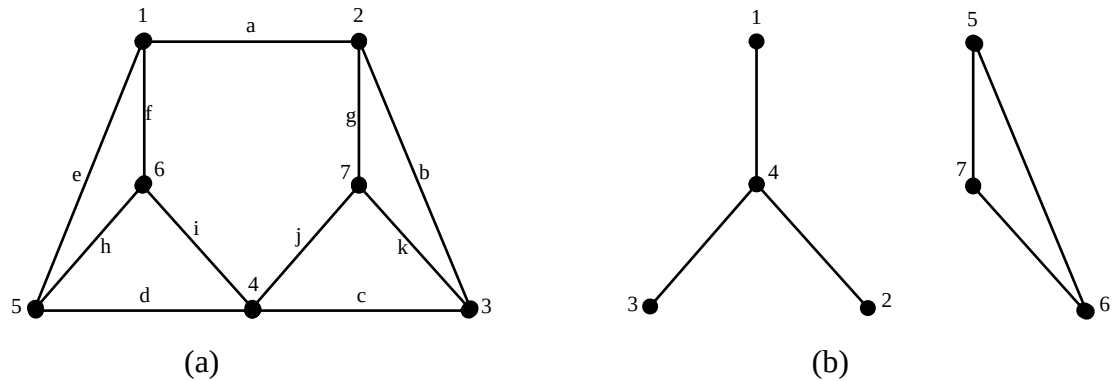


Figure 2.6: (a) A connected graph, and (b) a disconnected graph

2.1.4 An Incidence and Adjacency Matrices

In the graph G , when the edge e joins or connects the two vertices 1 and 2, these two vertices are called the *adjacent*. In the same time, both 1 and 2 are called the *incident* to the edge e . Likewise, in the graph G , two edges are adjacent if they have a vertex in common as it is shown in Figure 2.7 below.

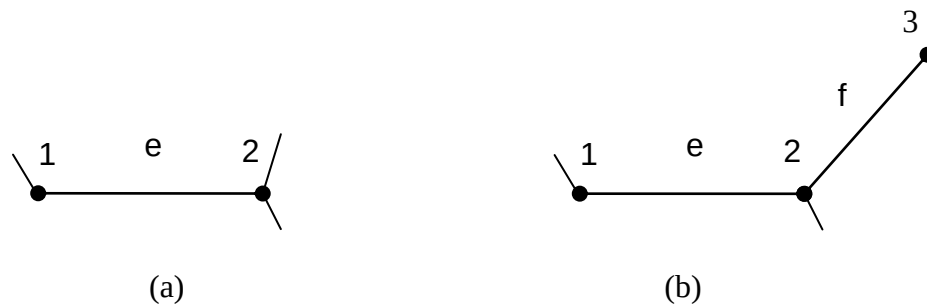


Figure 2.7: (a) Adjacent vertices, and (b) adjacent edges

Thus, based on the previous description, a pertinent question is raised. Are the graphs, as they are in diagram form, suitable for the computer calculations or the mathematical models to study their properties? The answer is no. Because of this, these two matrices are considered critical matrices. They are the incidence matrix and the adjacency matrix. Both of them are associated with the graph.

- The incidence matrix of a graph G is $n \times m$ matrix $M_G := (m_{ve})$, where m_{ve} is the number of times (0,1, or 2) that a vertex v and an edge e are incident.
- The adjacent matrix of a graph G is $n \times n$ matrix $A_G := (a_{uv})$, where a_{uv} is the number of edges that join vertices u and v .

Both the incidence and the adjacency matrices of the graph G in Figure 2.6 (a) can be built as follows:

$$\text{The incidence matrix; } M_{7 \times 11} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\text{The adjacency matrix; } A_{7 \times 7} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

It is easily apparent to notice that the adjacency matrix is much smaller than the incidence matrix because usually graphs have more edges than vertices. This causes the adjacency matrix to require less storage computer capability which is preferred in the computer programs where the consensus algorithm is applied [10].

2.2 Consensus Theory

As mentioned, the concept of the consensus is an area of interest from many researchers in different fields of a study. To illustrate the idea of the consensus, let us consider a system of three decision-making units where each unit can be known as an agent, or a generator as shown in Figure 2.4. Each agent has two parts: first part is the local controller and the second part is the consensus manager. The local controller collects system data and reports its own condition, or state, to the consensus manager. The mission of the consensus manager is to analyze this information and negotiates it with other connected neighborhood agents via the communication system. As result, the consensus manager will pass the decision back to the local controller. The local controller will update its situation depending on the received information and report back its new condition to the consensus manager again in a reciprocal, or back-and-forth process.

2.2.1 Consensus Definition

In a network of multi-agents, a “**Consensus**” means to reach an agreement regarding a certain quantity of an interest, depending on the state of all agents. A “**Consensus Algorithm**”, or a protocol, is an interaction rule that specifies the information exchange between an agent and all its neighbors in the same network [2].

2.2.2 Consensus in Networks

The topology of the system in Figure 2.4 can be represented using undirected graph $G = (V, E)$ where $V = \{1, 2, \dots, n\}$ and $E \subseteq V \times V$, whereas this graph can be described as using the adjacency matrix A as mentioned prior, of a finite system of n nodes. The adjacency matrix A is $n \times n$ matrix that depends on the number of the agents. The off-diagonal entry a_{ij} is the number of edges from node i to node j . If $a_{ij} \neq 0, i \neq j$

then the agent i^{th} is communicating with the agent j^{th} . In our case of a simple finite graph, the adjacency matrix is only a $(0, 1)$ -matrix with zeros on its diagonal. The neighbors of i^{th} agent are denoted by: $\mathcal{N}_i = \{j \in V : (i, j) \in E\}$.

For our system, the i^{th} node has $x_i \in R$ that denotes the state value it possess. The state x_i of a node can be a physical quantity, such as voltage, current, power, incremental cost, et cetera. It can be said that the system nodes, generators, or agents, have reached the consensus state if, and only if, the state value of i^{th} node is equal the state value of j^{th} node for all i, j [3]. i.e

$$x_i = x_j \text{ for all } i, j \quad (2.1)$$

The information discovery process for agent i is presented as follows:

$$x_i^{k+1} = x_i^k + \sum_{j \in \mathcal{N}_i} a_{ij} (x_j^k - x_i^k) , \quad i = 1, 2, \dots, n \quad (2.2)$$

Where:

x_i^{k+1}, x_i^k : are the discovered information by agent i (like average net power)

$\mathcal{N}_i$: is the neighbors that are connected to the agent i

a_{ij} : is the coefficient for information exchange between neighboring agents i and j

$$a_{ij} = \begin{cases} 0 < a_{ij} < 1 & \text{if agent } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

The information discovery process for the whole system is formed using matrix format as follows:

$$X^{k+1} = X^k + A * X^k \quad (2.3)$$

$$X^{k+1} = (I + A)X^k \quad (2.4)$$

$$X^{k+1} = DX^k \quad (2.5)$$

Where:

$$X^k = [x_1^k, \dots, x_i^k, \dots, x_n^k]$$

$$D = \begin{bmatrix} 1 - \sum_{j \in N_1} a_{1j} & \dots & a_{1i} & \dots & a_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{i1} & \dots & 1 - \sum_{j \in N_i} a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & \dots & a_{ni} & \dots & 1 - \sum_{j \in N_i} a_{ij} \end{bmatrix}$$

The D matrix is called the information discovery matrix, or sparse iteration matrix, for the system [18].

The coefficients a_{ij} in matrix D have to be properly designed, because they play a critical role in reaching consensus of a system. For instant, a fully-connected system can reach consensus in one step if a_{ij} is selected to be equal $\frac{1}{n}$ where n is the number of the system nodes. However, most of the systems are usually not fully-connected where they can suffer from loss of their connection lines.

Different methods are used to compute matrix D based on computing the coefficients a_{ij} . Some of those methods are summarized in the following:

- The Uniform Method: the weights are fixed and computed according the equation (2.6). The drawbacks of this method are the loss of the adaptivity and the slow convergence.

$$a_{ij} = \begin{cases} \frac{1}{n} & j \in \mathcal{N}_i \\ 1 - \sum_{j \in \mathcal{N}_i} \frac{1}{n} & i = j \\ 0 & otherwise \end{cases} \quad (2.6)$$

- The Metropolis Method: it is an improved method to accommodate the changes of system configuration which guarantees the stability of the information discovery

process. The weights are updated according the equation (2.7) below.

$$a_{ij} = \begin{cases} \frac{1}{[\max(n_i, n_j) + 1]} & j \in \mathcal{N}_i \\ 1 - \sum_{j \in \mathcal{N}_i} \frac{1}{[\max(n_i, n_j) + 1]} & i = j \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

- The Mean Metropolis Method: a new improved method that is presented in [18].

It provides the stability and the adaptivity for the information discovery process.

Their weights are updated according the equation (2.8) below.

$$a_{ij} = \begin{cases} \frac{2}{[n_i + n_j + \varepsilon]} & j \in N_i \\ 1 - \sum_{j \in N_i} \frac{2}{[n_i + n_j + \varepsilon]} & i = j \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

Where:

ε : is a very small number, and can be zero for large complex systems.

2.3 Consensus Algorithm Applications

Here are some applications where the consensus algorithm plays an important role:-

- Flocking Theory: A group of moving agents that have sensing and communicating devices in a large environment. The mission of the consensus algorithm in here is to make every agent reach the same velocity of its neighbors. The system of the network is a dynamic system with a dynamic topology. The topology is an agent state dependence, and it is computed locally for each agent. A good example for this is a group of non-pilot planes [2][12].
- Rendezvous in Space: In this case, the position of the agent is very important which makes the system an interactive position topology. This application is easier than the Flocking Theory because it does not require an agent-to-obstacle

clash avoidance [13].

- **Fast Consensus in Small-Worlds:** Using semi-definite convex programming, where the weight of the network is considered, leads to a slight increase in algebraic connectivity of the network, where the speed of the convergence of the consensus algorithms is measured. By keeping the network weights fixed, the topology can be designed to achieve a relatively high algebraic connectivity.
- **Distributed Sensor Fusion in Sensor Networks:** The above applications are considered as distributed sensor fusion. Distributed averaging problems require implementing the Kalman filter. The average of the inputs in the sensor networks is dynamically computed [14].
- **Distribution Formation Control:** In multivehicle systems, the design and analysis framework of distributed formation controllers uses vectors of relative positions of neighboring vehicles. In order the agents, as multivehicle, to move in formation, the agents have to cooperate, which requires consent and collaboration of every agent in the formation. It is clearly a consensus problem [16].
- **Economic Dispatch and Load Restoration Problems in Power System:** a conventional centralized control problem can be solved in a distribution manner by applying the consensus theory. Surely, this requires to choose the appropriate measures as a consensus variables [5][8][18]. More details are presented in the next chapter.

3. Economic Dispatch and Load Restoration Problems

3.1 Economic Dispatch Problem

The main target here is to get perfect use of serving equipment as much as possible with parallel to the highest financial benefits. In a power system, the Economic Dispatch is an optimal operation point of each power generator where the demand power is met, but the total generation cost is the lowest [6].

Several methods are used to solve the Incremental Cost equations, like setting value of Lambda and solving for generation values, or solving the value of Lambda directly for a specific load demand with different algorithms. Modern techniques use linear and non-linear programming like Piecewise-Linear fuel–Cost functions and Quadratic programming of the cost function.

Usually, a cost fuel curve of any power generator is formulated by quadratic function as follows [7]:

The fuel cost equation:

$$C_i(P_{Gi}) = \alpha_i + \beta_i P_{Gi} + \gamma_i P_{Gi}^2 \quad (3.1)$$

The Total fuel cost equation:

$$C_{Total} = \sum_{i=1}^n C_i(P_{Gi}) \quad (3.2)$$

Under the balance condition:

$$P_D - \sum_{i=1}^n P_{Gi} = 0 \quad (3.3)$$

Where:

C_i : the fuel cost of the i^{th} generator

$\alpha_i, \beta_i, \gamma_i$: the cost coefficients of the i^{th} generator

P_D : the total demanded power

P_{Gi} : the generated power of the i^{th} generator

The definition of the Incremental Cost IC for each generator is the same as the conventional economic dispatch:

$$IC_i = \frac{\partial C_i(P_{Gi})}{\partial P_{Gi}} = \lambda_i \quad i = 1, 2, \dots, n \quad (3.4)$$

By using the discrete-time consensus algorithm form then the Incremental Cost by using the first order discrete consensus algorithm is as follows [5]:

$$\lambda_i[k+1] = \sum_{j=1}^n d_{ij} \lambda_j[k] , \quad i = 1, 2, \dots, n \quad (3.5)$$

Where:

d_{ij} : is the (i, j) entry of the row-stochastic matrix D_n depends on the Laplacian matrix

$\lambda_i[k+1]$: updating IC of i^{th} takes in account the connected j^{th} generator to the i^{th} generator

In order to include the power balance constraints in the equation (3.3), the power balance error ΔP is defined as follows:

$$\Delta P = P_D - \sum_{i=1}^n P_{Gi} \quad (3.6)$$

Finally, the updating IC of the leader becomes as follows:

$$\lambda_i[k+1] = \sum_{j=1}^n d_{ij} \lambda_j[k] + \varepsilon \Delta P , \quad i = 1, 2, \dots, n \quad (3.7)$$

Where:

ε : is a positive scalar, and it is called the convergence coefficient that controls the convergence speed of the leader generator or the leader agent.

In the case where the power generation constraints are applied, the following conditions must be taken in account in the algorithm [5][7][8].

$$\begin{cases} P_{Gi} = P_{Gi\_max} , when P_{Gi} > P_{Gi\_max} \\ P_{Gi} = P_{Gi} , when P_{Gi\_min} \leq P_{Gi} \leq P_{Gi\_max} \\ P_{Gi} = P_{Gi\_min} , when P_{Gi} < P_{Gi\_min} \end{cases}$$

3.2 Load Restoration Problem

One of the consequences, after the protection system gives its orders to the circuit breakers to open, in order to isolate a faulty area in the power network, some un-faulted loads may trip and become out of service. Returning these loads back to the service as fast as possible, is one of the important aspects in the power networks. This action takes into account priority and capacity of each load. This procedure (returning un-faulted loads back to the service) is called in the power system the “*Load Restoration*” [18][19].

This kind of control task has been achieved in past years by using different computing algorithms, like genetic algorithm, particle swarm optimization, et cetera. A major drawback of these methods appears to be their need to a powerful central controller in order deal with a vast amount of information throughout the network. This leads to an extra cost and suffers from a single-point failure besides other drawbacks of a central controller that were mentioned earlier. In absence of question, solving the load restoration problem in a distribution control scheme is an intelligent solution to overcome all central controllers’ drawbacks.

Recently, Multi-Agent System (MAS) is an area of the interest, and different algorithms are applied to solve the load restoration problem. Some of them work with only radial power structures; others work with ring power structure. One advanced algorithm is presented in the reference [18]. It is capable to work even with a complex power structure, i.e., radial, ring, or mixed, because the proposed algorithm is independent of a system configuration.

In this case, the total net power of each agent is required for the calculation, and during stable normal operation the average of the total net power must be equal to zero.

$$X_{eq} = \frac{\sum_{i=1}^n x_i^o}{n} = 0, \quad i = 1, 2 \dots n \quad (3.8)$$

Where:

X : is discovered information which is the total net power of the agent i^{th}

n : is the total number of the agents

The overall information updating process i.e., the total average net power for each agent, is modeled the same as the equation (3.7) but without the balance error dimension.

$$X[k + 1] = DX[k] \quad (3.9)$$

Where:

D : is the sparse iteration matrix.

During a post fault period, the total net power equation (3.8) cannot be fulfilled because of the negative power shortage from the un-faulted agents, or loads. This requires immediate load restoration procedure. The Optimal load restoration decision can be obtained by satisfying coming two conditions:

1. Maximizing the capacity of restored loads and assuring higher priority loads be served first, i.e.,

$$\max \sum_{i=1}^N P_i \cdot P_{Li} \quad (3.10)$$

Where:

P_i : is the priority index associate with i^{th} load

P_{Li} : is the amount of power for an agent i^{th} needs to be restored

2. Sum of the total loads to be restored must not exceed the total generated power,

i.e.,

$$\sum_{i=1}^N P_{Gi} - \sum_{i=1}^N P_{Li} \geq 0 \quad (3.11)$$

Where:

P_{Gi} : is the amount of generated power for an agent i^{th} .

Identifying lost loads to be restored can be easily done by using the *0-1 Knapsack Problem*, which is a combinatorial optimization problem. Having a group of items where each item has its own weight and corresponding value, and finding the number of each item to be added in a collection with a condition of the total collection weight, is less than or equal to a given limited amount but the total value is as large as possible. Its mathematical model is summarized as follows [21]:

$$\begin{cases} \max f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n p_i x_i \\ \sum_{i=1}^n w_i x_i < A \\ x_i \in \{0,1\} \end{cases} \quad (3.12)$$

Where:

p_i : is the value of an agent i^{th}

w_i : is the weight of an agent i^{th}

A : is the limited total weight of the collection

The 0-1 knapsack problem can identify restored loads based on three factors, greedy by profit, or value, greedy by weight, or greedy by profit density. Since these loads are identified, the consensus algorithm will be activated to achieve a power balance again where the total net power for each system's agent should converge to zero.

3.3 Estimation Information Matrix in the Consensus Algorithm

Instead of computing the information matrix D in equations (3.5) and (3.9) based on the calculation methods that are presented in the chapter two, the estimation adaptive

techniques are applied to estimate it. In this thesis, Matlab codes were implemented based on the Gradient Algorithm (GA) and the Recursive Least Square Algorithm (RLS) to solve both the economic dispatch and the load restoration problems. The final forms of those estimation methods are presented in the following sections. The Complete mathematical derivation for both estimation algorithms is presented in the appendix A.

3.3.1 Gradient Algorithm Estimation Form

$$\lambda_i(k+1) = \hat{d}_{ii}\lambda_i(k) + \sum_{j \in \mathbb{N}_i} \hat{d}_{ij} [\lambda_j(k) - \lambda_i(k)] \quad (3.13)$$

$$\hat{d}_{ii} = 1 - \sum_{j \in \mathbb{N}_i} \hat{d}_{ij} \quad , \quad i = 1, 2, \dots, n \quad (3.14)$$

Inspired by the adaptive control theory, we propose the following cost function to be minimized [22].

$$\min_{\hat{d}_{ij}} \frac{1}{2} [\lambda_i(k+1) - \bar{\lambda}_i(k+1)]^2 \quad (3.15)$$

$$\hat{d}_{ij}(k+1) = \hat{d}_{ij}(k) - \mu [\lambda_i(k+1) - \bar{\lambda}_i(k+1)] \frac{\partial \lambda_i[k+1]}{\partial \hat{d}_{ij}} \quad (3.16)$$

$$\hat{d}_{ij}(k+1) = \hat{d}_{ij}(k) - \mu [\lambda_i(k+1) - \bar{\lambda}_i(k+1)] [\lambda_j(k) - \lambda_i(k)] \quad (3.17)$$

$$\hat{d}_{ij}(k+1) = \hat{d}_{ij}(k) - \mu \varphi_i(k) [\lambda_i(k+1) - \bar{\lambda}_i(k+1)] \quad (3.18)$$

$$\varphi_i(k) = [\lambda_j(k) - \lambda_i(k)] \quad j = 1, 2, \dots, \mathbb{N}_i \quad (3.19)$$

In the equation (3.18), $\bar{\lambda}_i$ is the average of the neighbors of a i^{th} node, and it is computed as follows:

$$\bar{\lambda}_i(k+1) = \frac{1}{n_i} \sum_{j \in \mathbb{N}_i} \lambda_j(k+1) \quad (3.20)$$

The initial values: $\hat{d}_{ij}(0) = 0$ $i \neq j$ and $\hat{d}_{ii}(0) = 1 - \sum_{j \in \mathbb{N}_i} \hat{d}_{ij}(0)$ $i \neq j$

$\mu = 0.5$, 0.1 or less is a scalar to increase the speed of the convergence.

3.3.2 Recursive Least Square Algorithm Estimation Form

$$\hat{d}_{ij}(k+1) = \hat{d}_{ij}(k) - P_i(k)\varphi_i(k)[\lambda_i(k+1) - \bar{\lambda}_i(k+1)] \quad (3.21)$$

$$\varphi_i(k) = [\lambda_j(k) - \lambda_i(k)] \quad j = 1, 2, \dots, \mathbb{N}_i \quad (3.22)$$

$$P_i(k) = P_i(k-1) + \frac{P_i(k-1)\varphi_i(k)\varphi_i(k)^T P_i(k-1)}{1 + \varphi_i(k)^T P_i(k-1)\varphi_i(k)} \quad (3.23)$$

In the equation (3.21), $\bar{\lambda}_i$ is the average of the neighbors of a i^{th} node, and it is computed as follows:

$$\bar{\lambda}_i(k+1) = \frac{1}{n_i} \sum_{j \in \mathbb{N}_i} \lambda_j(k+1) \quad (3.24)$$

The initial values: $\hat{d}_{ij}(0) = 0 \quad i \neq j$ and $\hat{d}_{ii}(0) = 1 - \sum_{j \in \mathbb{N}_i} \hat{d}_{ij}(0) \quad i \neq j$

$P_i(0) = P_i I$, $P_i > 0$ is a scalar matrix to increase the speed of the convergence.

4. Study Cases and Results

To verify our new estimation methods of computing average consensus, in both cases, the Economic Dispatch and the Load Restoration problems, they were tested on cases that had been studied before. We compared our results with theirs to make sure the performance of our new estimation algorithms gave the same correct results of the previous studies.

4.1 Three Generators System for Economic Dispatch Problem

The system contains three generators, or agents, which provide power supply to an electric load P_D . The system parameters are summarized in the table 4.1 below [7]:

Table 4.1: Three generators; system parameters

Agent Or Generator	Cost Coefficients used in equation (3.1)			Power initial [MW]
	α [\$/hr]	β [\$/MWhr]	γ [\$/MW ² hr]	
1 st	561	7.92	0.001562	450
2 nd	310	7.85	0.001940	300
3 rd	078	7.97	0.004820	100

In this case, the new estimation method is applied to solve the economic dispatch problem. Here, instead of computing the row- stochastic matrix D , based on the Laplacian Matrix L , the gradient algorithm is used to estimate it. The communication topology of this system is the same as Figure 1.1 (b). The algorithm will solve the economic dispatch problem so that the Incremental Cost of each generator converges to the same optimal value. The next sections show our results based on the estimation method.

4.1.1 Three Generators System with New Estimation Method using Gradient Algorithm without Power Constraints

In this case, there were no produced power limits for the system generators. The situation's summary is as follows:

- 1- The system parameters were taken from the table 4.1
- 2- The row- stochastic matrix D was estimated by using the Gradient Algorithm
- 3- The required demanded power was 850 MW and 950 MW
- 4- The convergence gradient gain coefficient $\mu = 0.5$

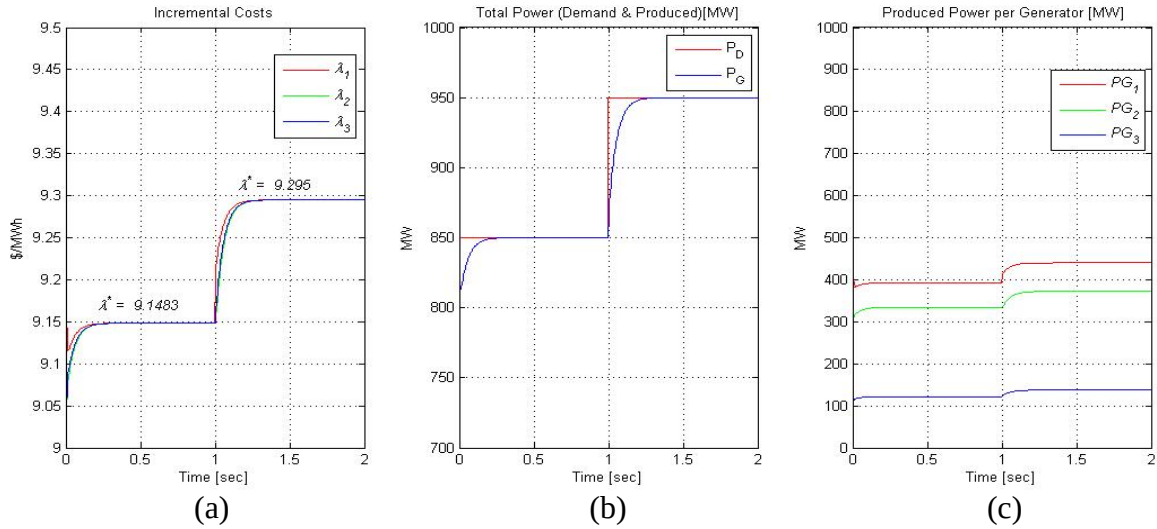


Figure 4.1: Three generators using GA; IC's and power (no power constraints)

As it is seen in Figure 4.1 (a), all the three Incremental Costs λ_1, λ_2 , and λ_3 converged very fast to the same optimal value $\lambda^* = 9.1483$ \$/MWh. After 1 second, the demanded power increased to 950 MW. We notice that the leader (agent 1), red line, responded faster than the others after the change happened, then it raised the other agents to a new Incremental Cost value, and after 1.4 sec they reached to the consensus again, which is a new optimal value $\lambda^* = 9.295$ \$/MWh. In both stages, the generated power reached the demanded power quickly. Because there were no power constraints, we notice that all agents produced power without any limits as it is shown in Figure 4.1 (c).

Also the power balance had been satisfied as it is shown in Figure 4.1 (b). Our results are exactly the same as the results in the reference [5] where the Incremental Cost was computed based on the calculation methods in equations (2.6), (2.7), and (2.8).

4.1.2 Three Generators System with New Estimation Method using Gradient Algorithm with Power Constraints

In this case, there were produced power limits for the system generators. The situation's summary is as follows:

1. The system parameters were taken from the table 4.1 except $\gamma^{(1)} = 0.00128$ \$/MW2hr so that agent one can produce more than 600 MW
2. The row- stochastic matrix D was estimated by using the Gradient Algorithm
3. The required demanded power was 850 MW and 950 MW
4. The convergence gradient gain coefficient $\mu = 0.5$
5. The generator maximum's limits were [600 400 200] MW
6. The generator minimum's limits were [150 100 050] MW

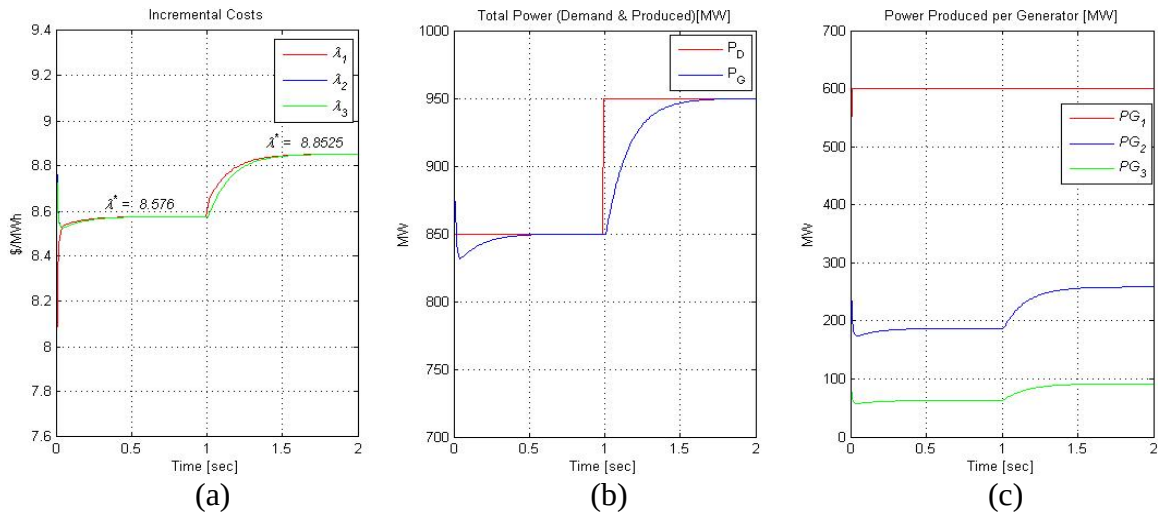


Figure 4.2 Three generators using GA; IC's and power (with power constraints)

In this case, the generators' power limits were applied in solving the Incremental Fuel Cost. As we see in Figure 4.2 (a), all the three Incremental Costs λ_1 , λ_2 , and λ_3

converged very fast to the same optimal value $\lambda^* = 8.576$ \$/MWh. After 1 second, the demanded power increased to 950 MW. We noticed that the leader (agent 1), red line, responded faster than the others after the change happened, then it raised the other agents to a new Incremental Cost value, and after 1.6 sec they reached to the consensus again, which is a new optimal value $\lambda^* = 8.852$ \$/MWh. In both stages, the generated power reached the demanded power quickly. Because of the power constraints, we notice that the generator one (the leader) stuck on 600 MW as a result of its power limits as shown in Figure 4.2 (c). Also, the power balance and the power constraints had been satisfied as shown in Figure 4.2 (b). Our results are exactly the same as the results in the reference [7] where the Incremental Cost was computed based on the calculation methods.

4.2 Five Generators System for Economic Dispatch Problem

The system contains five generators or agents provide power supply to an electric load P_D . The system parameters are summarized in the table 4.2 below [7]:

Table 4.2: Five generators; system parameters

Agent or Generator	Cost Coefficients used in equation (3.1)			Power initial [MW]
	α [\$/hr]	β [\$/MWhr]	γ [\$/MW ² hr]	
1 st	561	7.92	0.001562	200
2 nd	310	7.85	0.001940	250
3 rd	078	7.97	0.004820	100
4 th	561	7.92	0.001562	200
5 th	078	7.80	0.004820	100

In this case, the gradient algorithm and the recursive least square algorithm are used to estimate the row-stochastic matrix D. The communication network of this system has different topologies as it is shown in Figure 4.3 below. The Incremental Cost algorithm will solve the economic dispatch problem so that the Incremental Cost of each generator converges to the same optimal value. The next sections show our results based on the estimation methods.

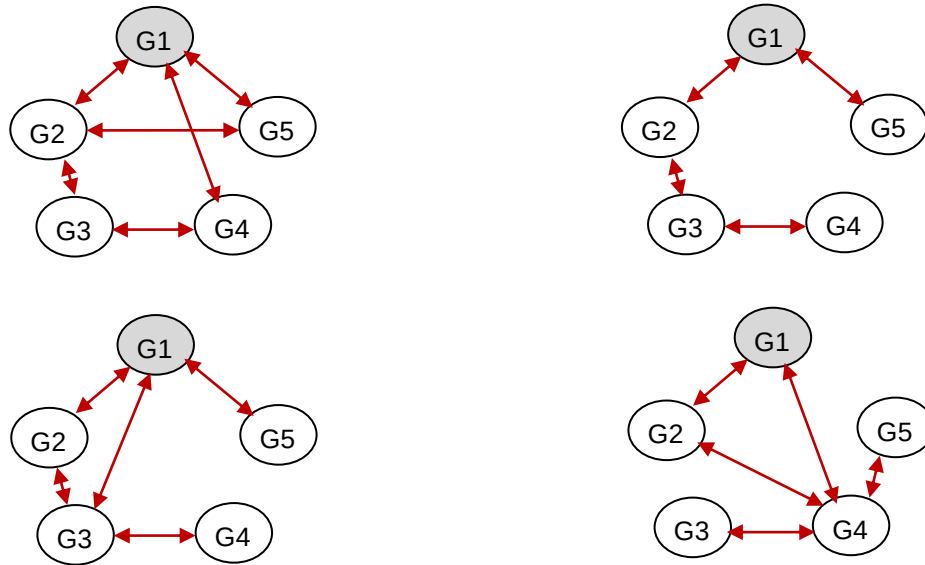


Figure 4.3: Five generators; different system topologies

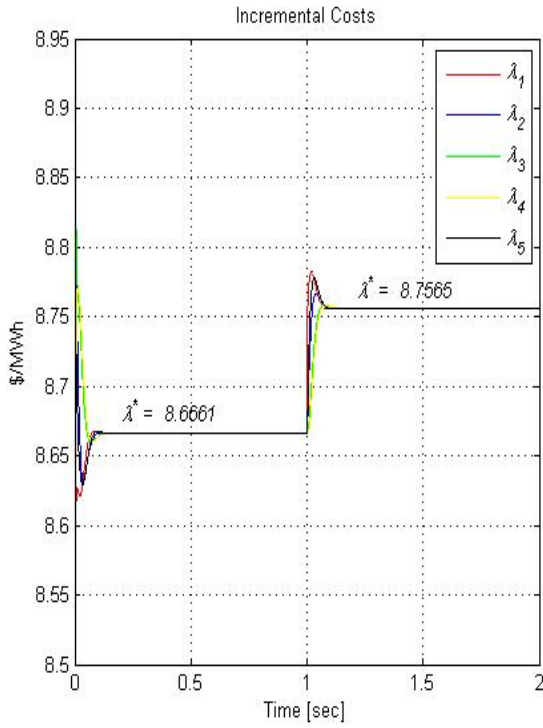
4.2.1 Five Generators with New Estimation Method using Gradient Algorithm

Three different cases for the initial values of the adjacency matrix for different system topologies were studied. The situation's summary is as follows:-

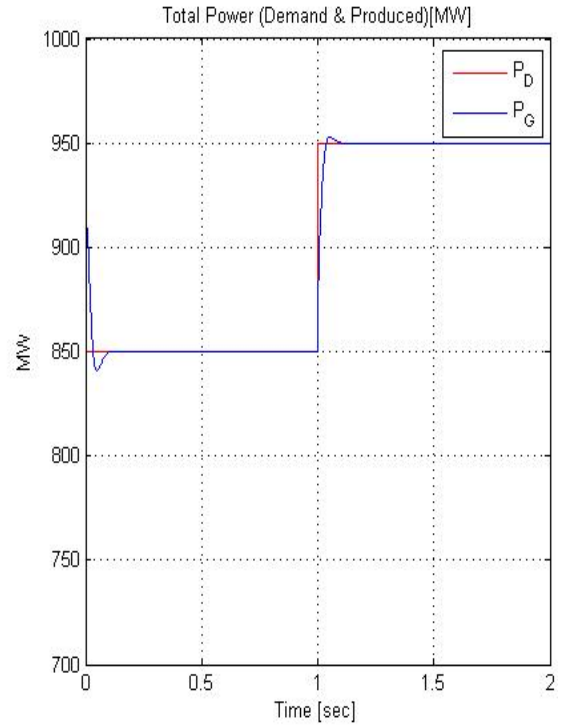
- 1- The system parameters were taken from the table 4.2
- 2- The row- stochastic matrix D was estimated by using the Gradient Algorithm
- 3- The required demanded power was 850 MW and 950 MW
- 4- The convergence gradient gain coefficient $\mu = 0.5$
- 5- Different system topologies were used as it is shown in Figure 4.3
- 6- No power constraints were applied

4.2.1.1 Five Generators using GA; Guessing Initial Values $d_{ij} = d_{ji} = 0.0$

$$d_{ij} = d_{ji} = 0.0, \quad d_{ii} = 1 - \sum_{j=1:N_i, i \neq j}^{N_i} d_{ij} \Rightarrow d_o = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$



(a)

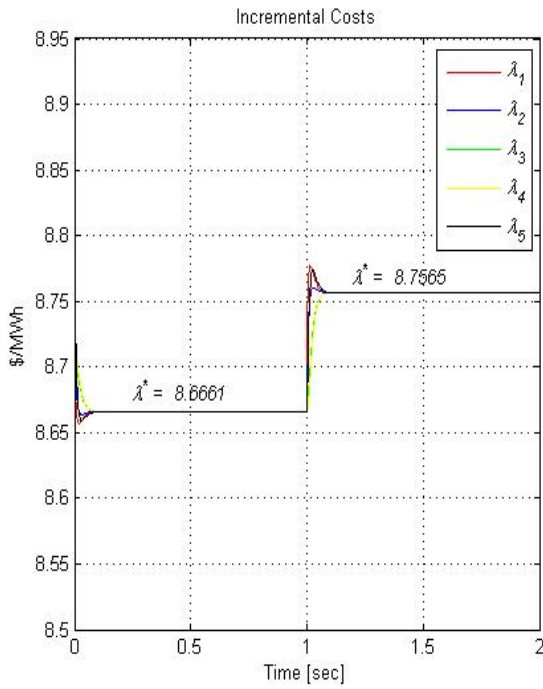


(b)

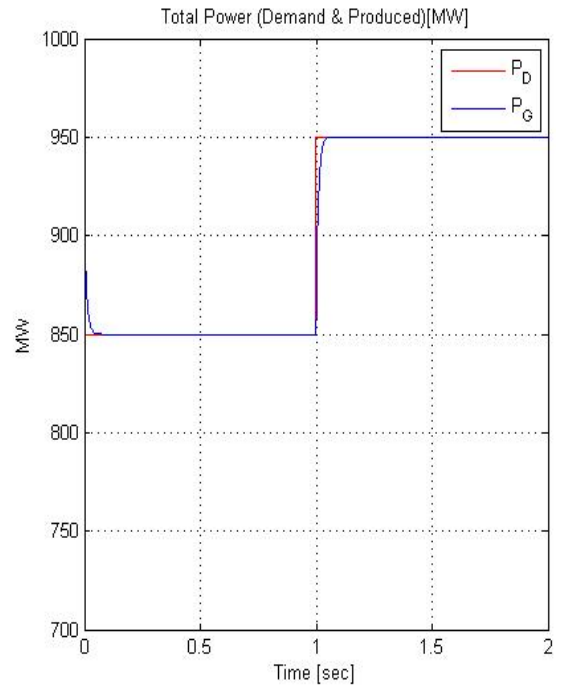
Figure 4.4: Five generators using GA; IC's and power ($d_{ij} = d_{ji} = 0.0$)

4.2.1.2 Five Generators using GA; Guessing Initial Values $d_{ij} = d_{ji} \neq 0.0$

$$d_{ij} = d_{ji} \neq 0.0, d_{ii} = 1 - \sum_{j=1:N_i, i \neq j}^{N_i} d_{ij} \Rightarrow d_o = \begin{bmatrix} 0.10 & 0.30 & 0.20 & 0.25 & 0.15 \\ 0.30 & 0.05 & 0.10 & 0.10 & 0.05 \\ 0.20 & 0.10 & 0.10 & 0.40 & 0.20 \\ 0.25 & 0.10 & 0.40 & 0.15 & 0.10 \\ 0.15 & 0.05 & 0.20 & 0.10 & 0.05 \end{bmatrix}$$



(a)



(b)

Figure 4.5: Five generators using GA; IC's and power ($d_{ij} = d_{ji} \neq 0.0$)

4.2.1.3 Five Generators using GA; Guessing Initial Values $d_{ij} \neq d_{ji} \neq 0.0$

$$d_{ij} \neq d_{ji} \neq 0.0, \quad d_{ii} = 1 - \sum_{j=1:N_i, i \neq j}^{N_i} d_{ij} \Rightarrow d_o = \begin{bmatrix} 0.30 & 0.30 & 0.15 & 0.05 & 0.20 \\ 0.15 & 0.20 & 0.15 & 0.30 & 0.20 \\ 0.20 & 0.10 & 0.05 & 0.40 & 0.25 \\ 0.20 & 0.20 & 0.30 & 0.15 & 0.15 \\ 0.15 & 0.05 & 0.20 & 0.10 & 0.50 \end{bmatrix}$$

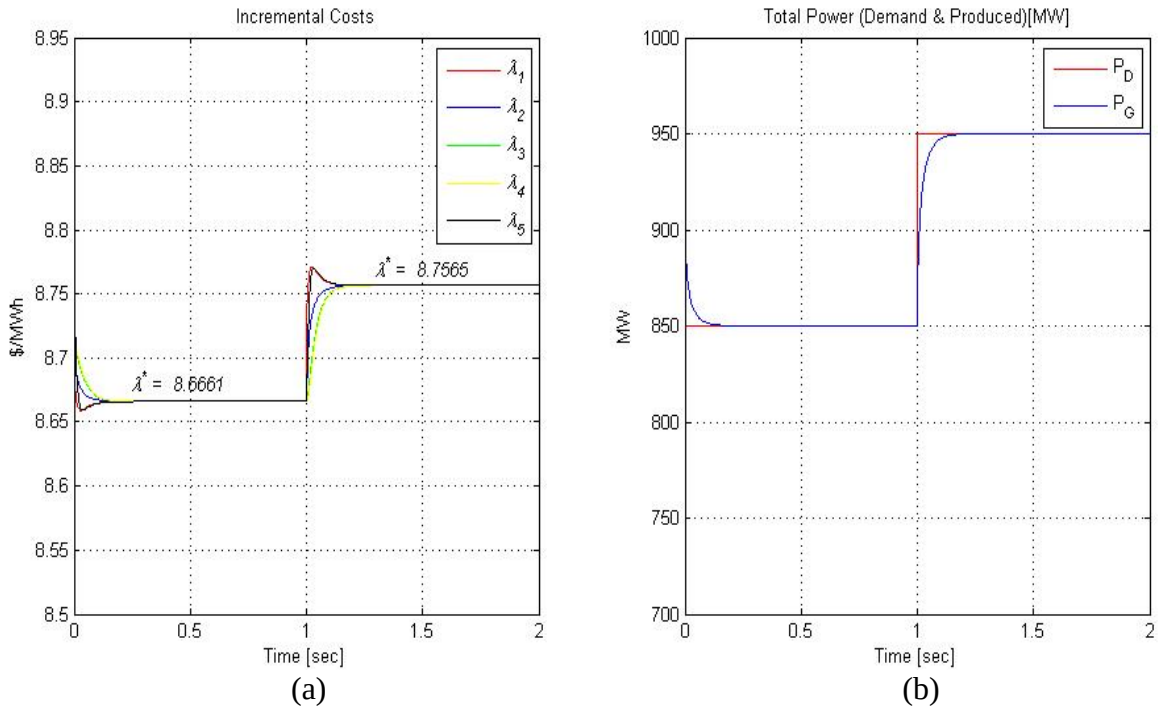


Figure 4.6: Five generators using GA; IC's and power ($d_{ij} \neq d_{ji} \neq 0.0$)

It can be seen in Figures 4.4 (a), 4.5 (a), and 4.6 (a) by applying the gradient algorithm, all five Incremental Costs $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and λ_5 converged very fast to the same optimal value $\lambda^* = 8.666$ \$/MWh. After 1 second, the demanded power increased to 950 MW. It is noticeable that the leader (agent 1), red line, responded faster than the others after the change happened, then it raised the other agents to a new Incremental Cost value, and after 1.1 sec they reached to the consensus again, which is a new optimal

value $\lambda^* = 8.756 \text{ \$/MWh}$. It can be easily noticed that in Figure 4.5(a) where $d_{ij} = d_{ji} \neq 0.0$, the agents reached the consensus faster than the other cases because of the symmetry of the initial matrix. Figures 4.4 (b), 4.5 (b), and 4.6 (b) show the total generated power by the agents. When the demanded power changed from 850 MW to 950 MW after 1 second of operation, all agents increased their produced power to meet the demanded power. Therefore, the power balance had been satisfied. Our results are exactly the same as the results in the reference [5] where the Incremental Cost was computed based on the calculation methods.

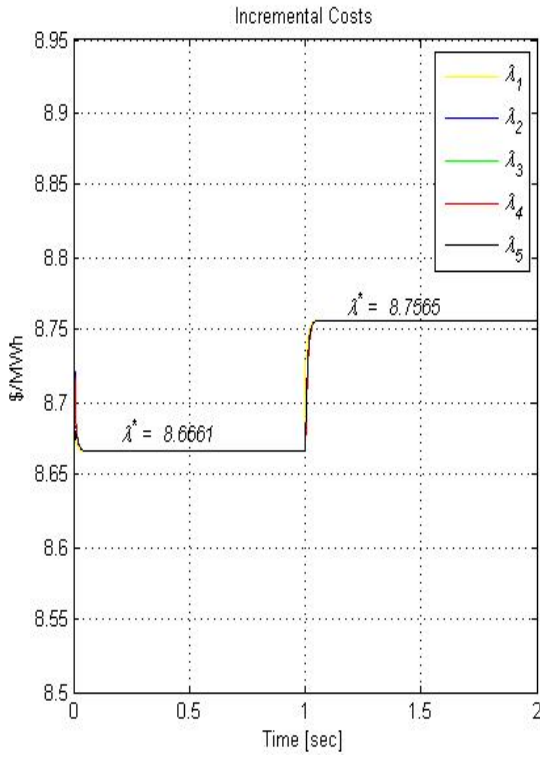
4.2.2 Five Generators System with New Estimation Method using Recursive Least Square Algorithm

Three different cases for the initial values of the adjacency matrix for different system topologies were studied. The situation's summary is as follows:-

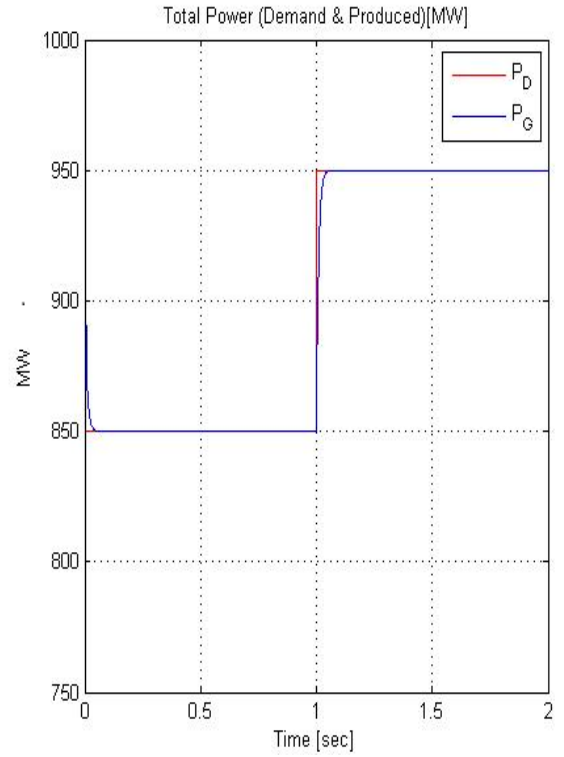
- 1- The system parameters were taken from the table 4.2
- 2- The row- stochastic matrix D was estimated by using the Gradient Algorithm
- 3- The required demanded power was 850 MW and 950 MW
- 4- Different system topologies were used as it is shown in Figure 4.3
- 5- No power constraints were applied
- 6- The Covariance Matrix's initial $p_o = 10I$

4.2.2.1 Five Generators using RLS; Guessing Initial Values $d_{ij} = d_{ji} = 0.0$

$$d_{ij} = d_{ji} = 0.0, \quad d_{ii} = 1 - \sum_{j=1:N_i, i \neq j}^{N_i} d_{ij} \Rightarrow d_o = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$



(a)



(b)

Figure 4.7: Five generators using RLS; IC's and power ($d_{ij} = d_{ji} = 0.0$)

4.2.2.2 Five Generators using RLS; Guessing Initial Values $d_{ij} = d_{ji} \neq 0.0$

$$d_{ij} = d_{ji} \neq 0.0, d_{ii} = 1 - \sum_{j=1:N_i, i \neq j}^{N_i} d_{ij} \Rightarrow d_o = \begin{bmatrix} 0.10 & 0.30 & 0.20 & 0.25 & 0.15 \\ 0.30 & 0.05 & 0.10 & 0.10 & 0.05 \\ 0.20 & 0.10 & 0.10 & 0.40 & 0.20 \\ 0.25 & 0.10 & 0.40 & 0.15 & 0.10 \\ 0.15 & 0.05 & 0.20 & 0.10 & 0.05 \end{bmatrix}$$

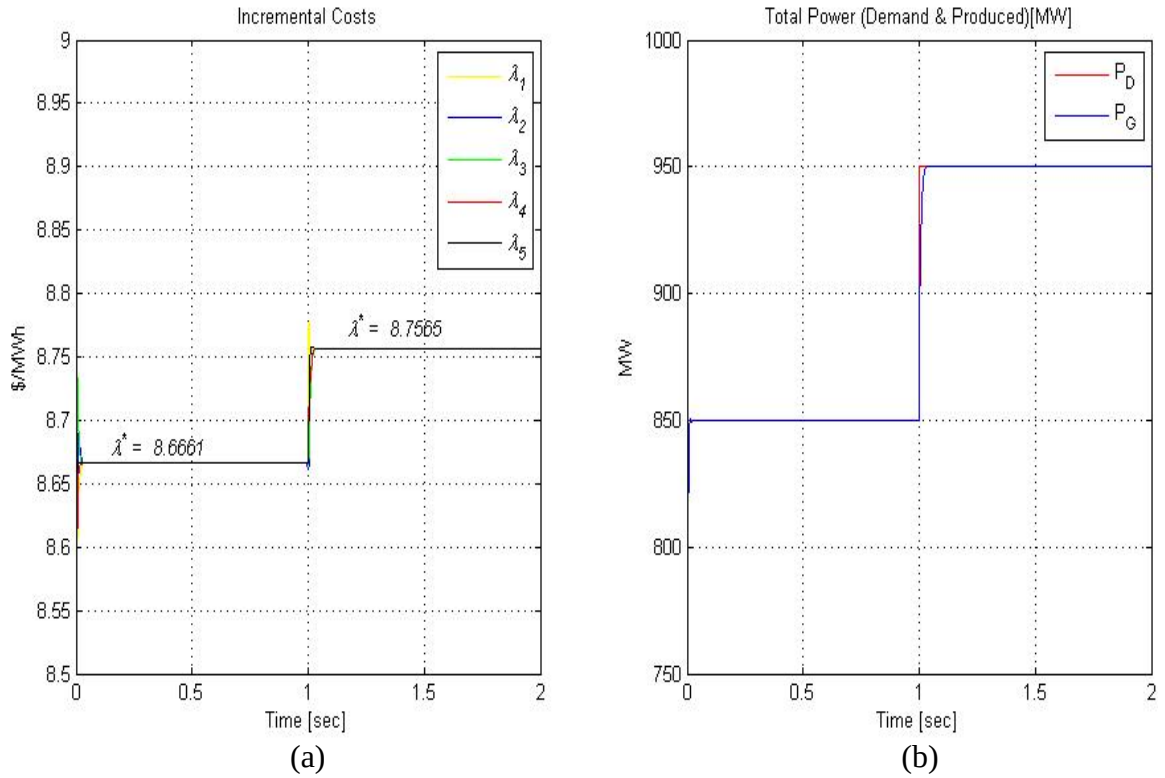


Figure 4.8: Five generators using RLS; IC's and power ($d_{ij} = d_{ji} \neq 0.0$)

4.2.2.3 Five Generators using RLS; Guessing Initial Values $d_{ij} \neq d_{ji} \neq 0.0$

$$d_{ij} \neq d_{ji} \neq 0.0, \quad d_{ii} = 1 - \sum_{j=1:N_i, i \neq j}^{N_i} d_{ij} \Rightarrow d_o = \begin{bmatrix} 0.30 & 0.30 & 0.15 & 0.05 & 0.20 \\ 0.15 & 0.20 & 0.15 & 0.30 & 0.20 \\ 0.20 & 0.10 & 0.05 & 0.40 & 0.25 \\ 0.20 & 0.20 & 0.30 & 0.15 & 0.15 \\ 0.15 & 0.05 & 0.20 & 0.10 & 0.50 \end{bmatrix}$$

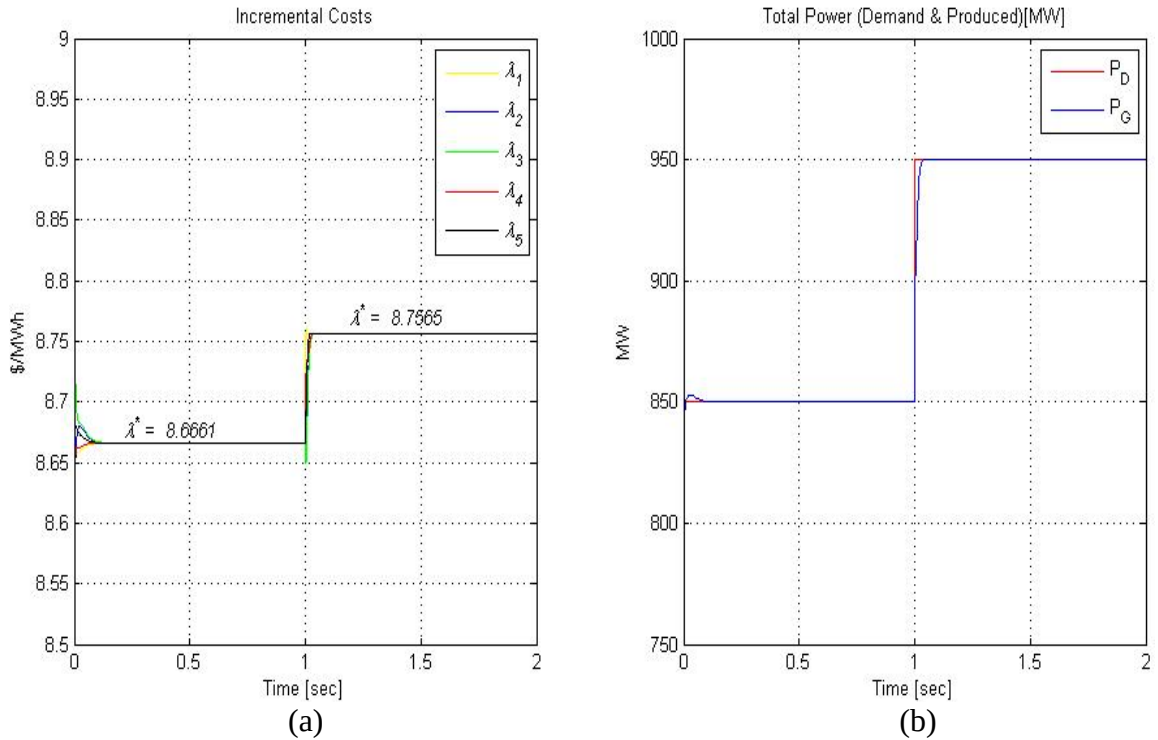


Figure 4.9: Five generators using RLS; IC's and power ($d_{ij} \neq d_{ji} \neq 0.0$)

It can be seen in Figures 4.7 (a), 4.8 (a), and 4.9 (a) by applying recursive least square algorithm, all five Incremental Costs λ_1 , λ_2 , λ_3 , λ_4 , and λ_5 converged very fast to the same optimal value $\lambda^* = 8.666$ \$/MWh. After 1 second, the demanded power increased to 950 MW. It was noticeable that the leader (agent 1), yellow line, responded

faster than the others after the change happened, then it raised the other agents to a new Incremental Cost value, and after 1.05 seconds they reached the consensus again, which is a new optimal value $\lambda^* = 8.756 \text{ \$/MWh}$. It can be easily noticed that in Figure 4.8(a) where $d_{ij} = d_{ji} \neq 0.0$, the agents reached the consensus faster than the other cases because of the symmetry of the initial matrix. Overall, the performance of RLS algorithm has smoother convergence to the optimal value than the GA algorithm, and this can be easily noticeable when their results are compared. Figures 4.7 (b), 4.8 (b), and 4.9 (b) show the total produced power by all agents. When the demanded power changed from 850 MW to 950 MW after 1 second of operation, all agents increased their produced power to meet the demanded power. So the power balance had been satisfied. Our results are exactly the same as the results in the reference [5] where the Incremental Cost was computed based on the calculation methods in equations (2.6), (2.7), and (2.8).

4.2.3 Five Generators with New Estimation Method using GA and RLS algorithms where the System Topology Changes with Time

In this case, the system topology changes with time. The situation's summary is as follows:-

1. The system parameters were taken from the table 4.2
2. The row- stochastic matrix D was estimated by using the Gradient Algorithm and the Recursive Least Square Algorithm with $d_0 = 0.0$ for both algorithms
3. The required demanded power was 850 MW and 950 MW
4. The system topology was changing with time as it is shown in Figure 4.10 on the next page
5. No power constraints were applied

The topology of the system changes with time as follows:-

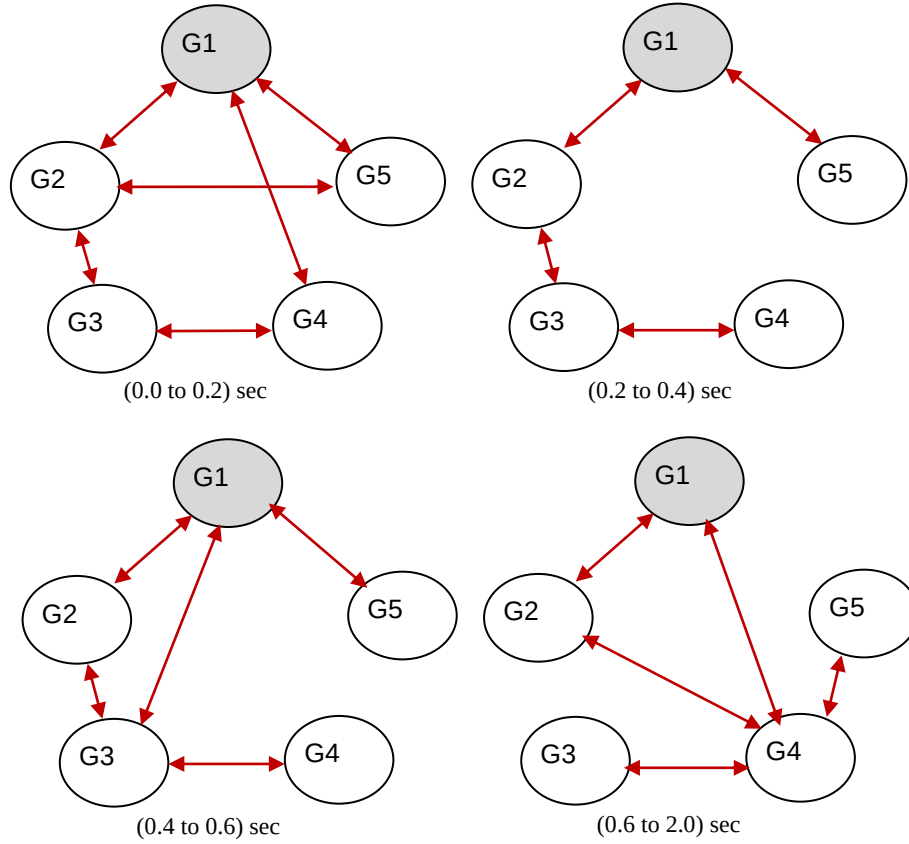


Figure 4.10: Five generators; changing sequence of the system topology

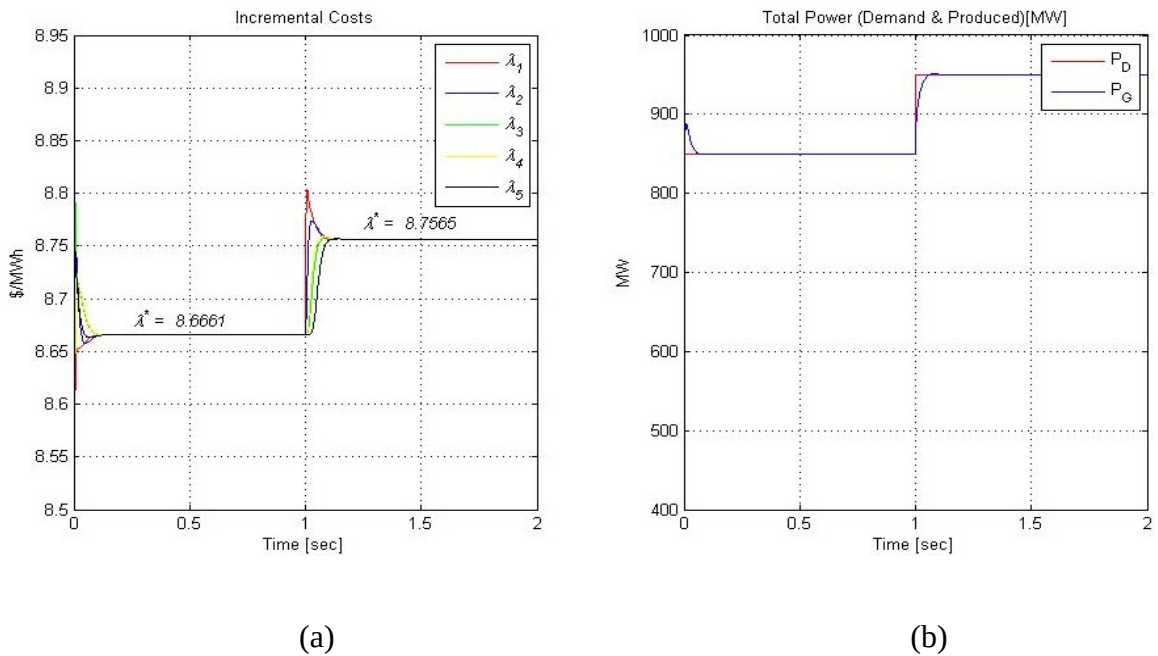


Figure 4.11: Five generators using GA; IC's and power (topology changes with time)

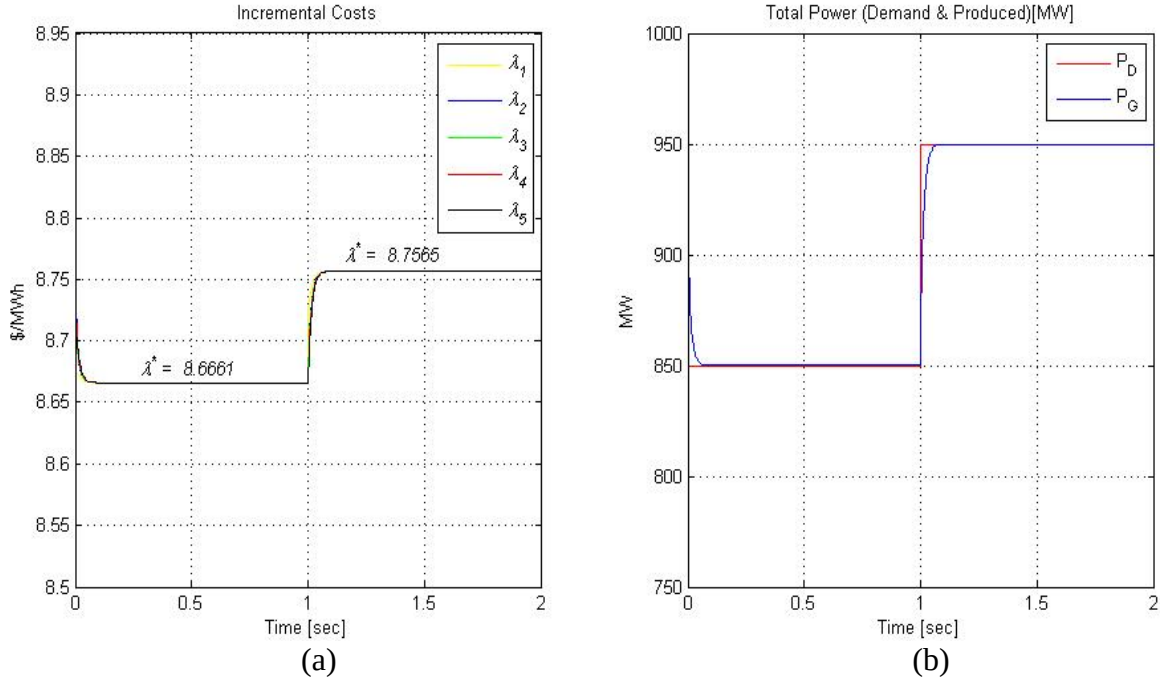


Figure 4.12: Five generators using RLS; IC's and power (topology changes with time)

It can be seen from Figures 4.11(a) and 4.12(a), by using the gradient algorithm and the recursive least square algorithm, all five Incremental Costs $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and λ_5 converged very fast to the same optimal value $\lambda^* = 8.666$ \$/MWh and $\lambda^* = 8.756$ \$/MWh corresponding to the demanded power from 850 MW to 950 MW. Figures 4.11(b) and 4.12 (b) illustrate the total generated power. It is noticeable that the total generated power increased to meet the demanded power. This case lead to the conclusion the connected agents reached the consensus even though their topology changed with time.

4.2.4 Five Generators with New Estimation Method using GA and Agents 4 and 5 are Become Disconnected

In this case, two agents became disconnected from the rest of the system. What will happen to the system convergence and the produced power? The changing sequence of the system topology is illustrated in Figure 4.13 on the next page. The agents 4 and 5 have been selected as the disconnected agents after 1.2 sec of operation. The situation's

summary is as follows:-

1. The system parameters were taken from the table 4.2
2. The row- stochastic matrix D was estimated by using the Gradient Algorithm
3. The required demanded power was 850 MW
4. The convergence gradient gain coefficient $\mu = 0.5$
5. No power constraints were applied.

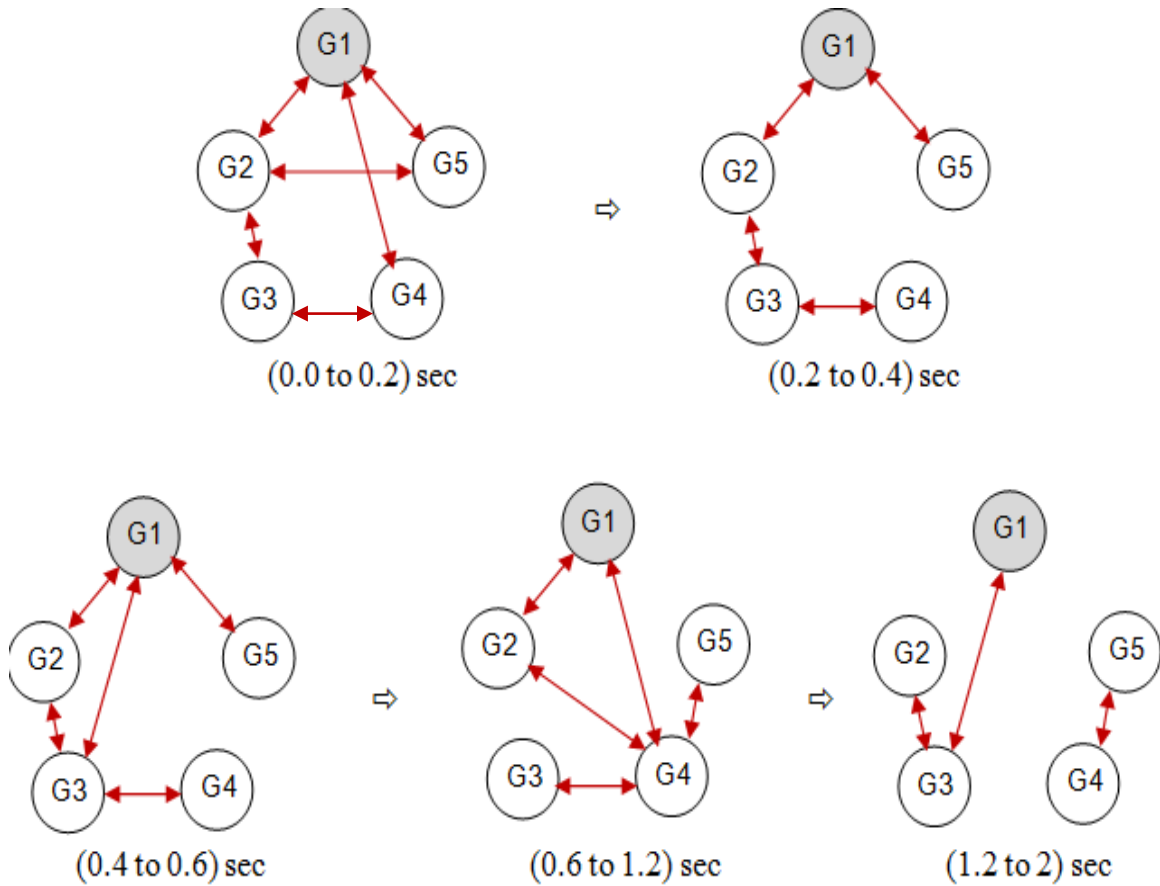


Figure 4.13: Five generators; agents 4 and 5 become disconnected

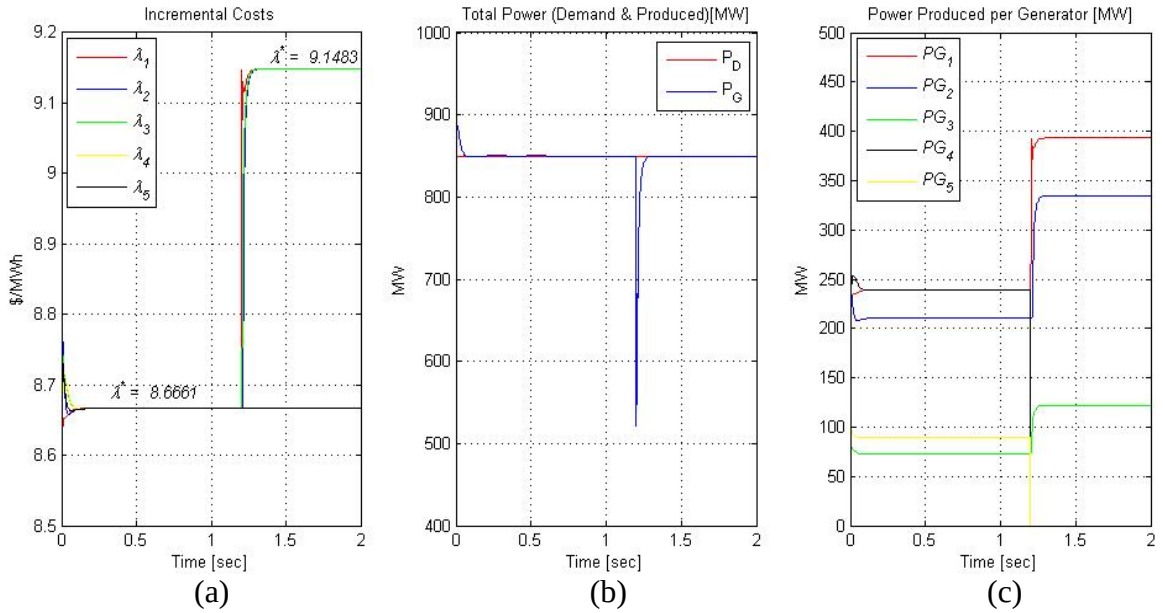


Figure 4.14: Five generators using GA; IC's and power (two agents 4 and 5 become disconnected)

It can be seen from Figure 4.14 (a), by using the gradient algorithm, all five Incremental Costs $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and λ_5 converged very fast to the same optimal value $\lambda^* = 8.666$ \$/MWh. After 1.2 sec of operation, two agents (4 & 5) became disconnected from the rest of the system while the demanded power still at 850 MW. It can be noticed that the other three agents (1, 2, and 3) converged to a new Incremental Cost value $\lambda^* = 9.148$ \$/MWh. As the system lost the connection with the disconnected agents so it kept the last Incremental Costs of them just before they became disconnected. Because the system lost part of the generated power from the disconnected agents, as it is seen in Figure 4.14 (b), the other agents had to compensate the shortage in the generated power, and this what exactly happened in a few milliseconds as it illustrated in Figure 4.14 (c). The agents 1, 2, and 3 increased their produced power to match the demanded power of the system. Finally, the power balance had been satisfied.

4.2.5 Five Generators with New Estimation Method using RLS and Check the Necessity of the Reaching Consensus

This case is to check the necessity that said “In order the agents to reach the consensus, the sum of all columns of the adjacency matrix d must be equal to one”, so the summation of all rows (red lines) and columns (blue lines) were plotted in order to check whether this necessity hold or not. The situation’s summary is as follows:-

1. The system parameters were taken from the table 4.2
2. The row- stochastic matrix D was estimated by using the RLS and $d_0 = 0.0$
3. The required demanded power was 850 MW
4. No power constraints were applied.

The system topology was changing with time as it is shown in Figure 4.15 below.

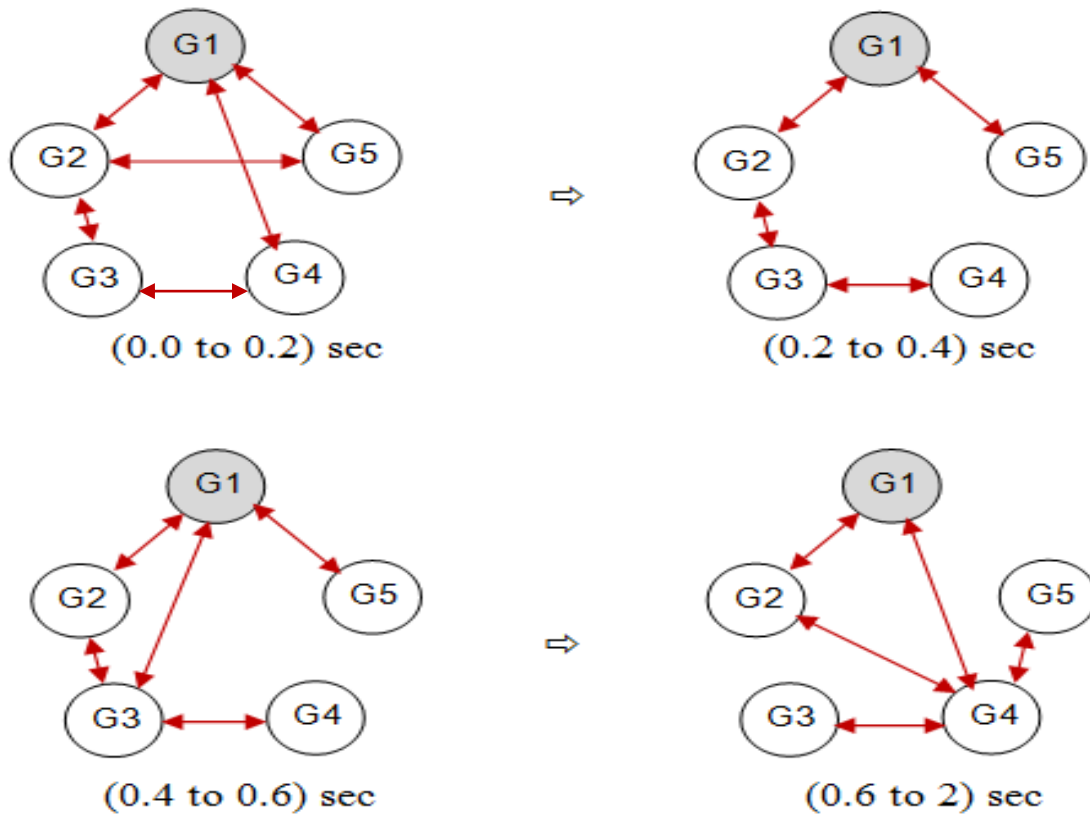


Figure 4.15: Five generators; changing sequence of system topology

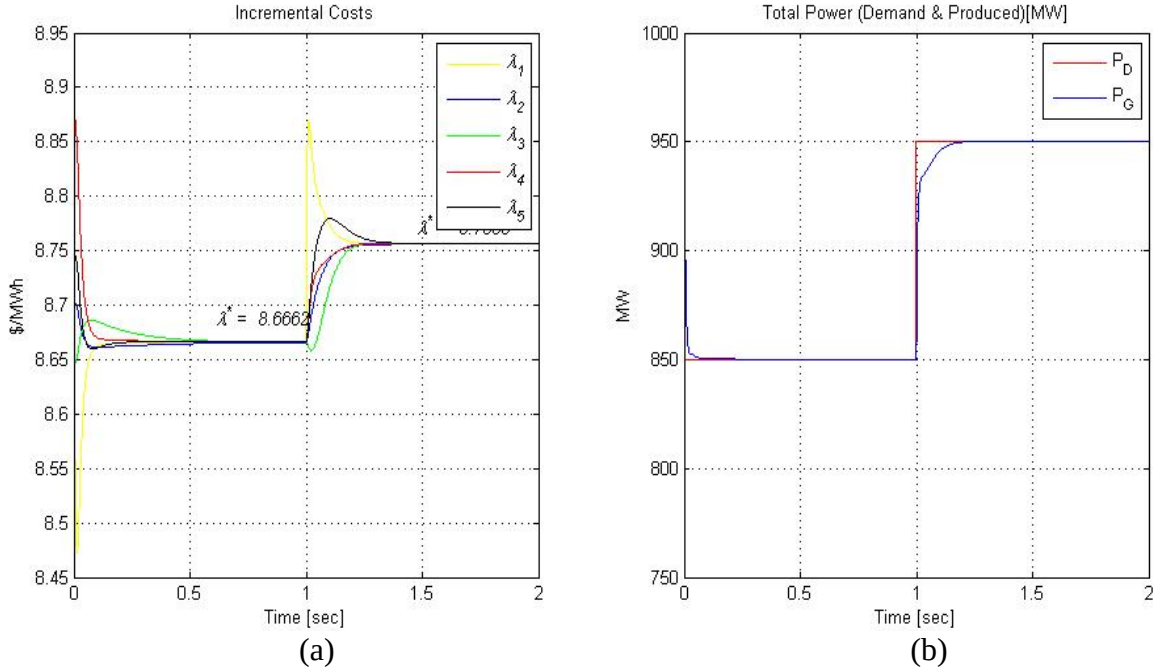


Figure 4.16: Five generators using RLS; IC's and power ($P_o = 0.01$)

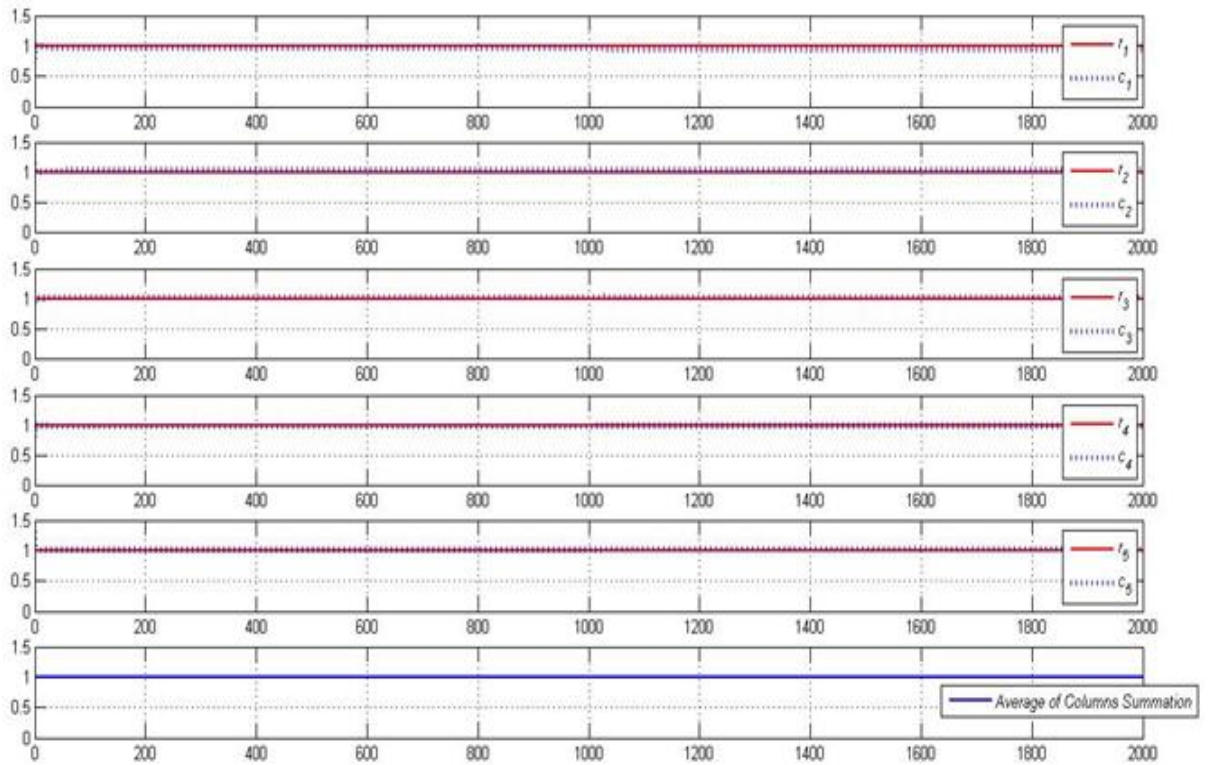


Figure 4.17: Five generators; adjacency rows and columns' summation ($P_o = 0.01$)

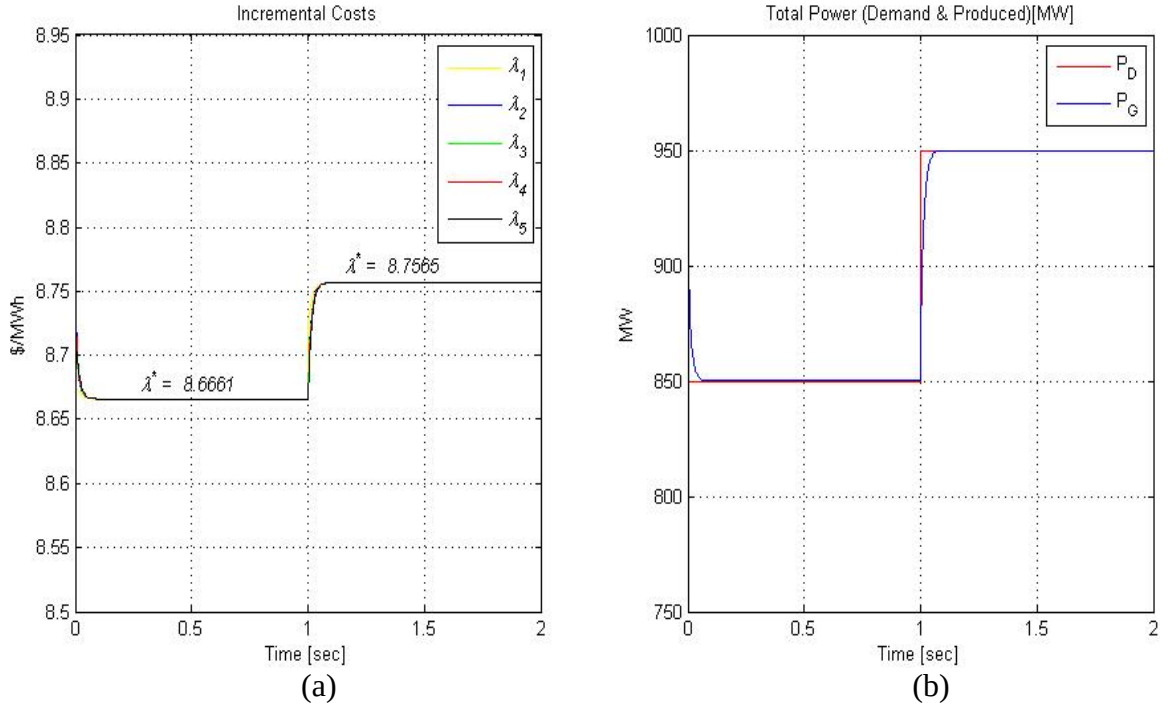


Figure 4.18: Five generators using RLS; IC's and power ($P_o = 10$)

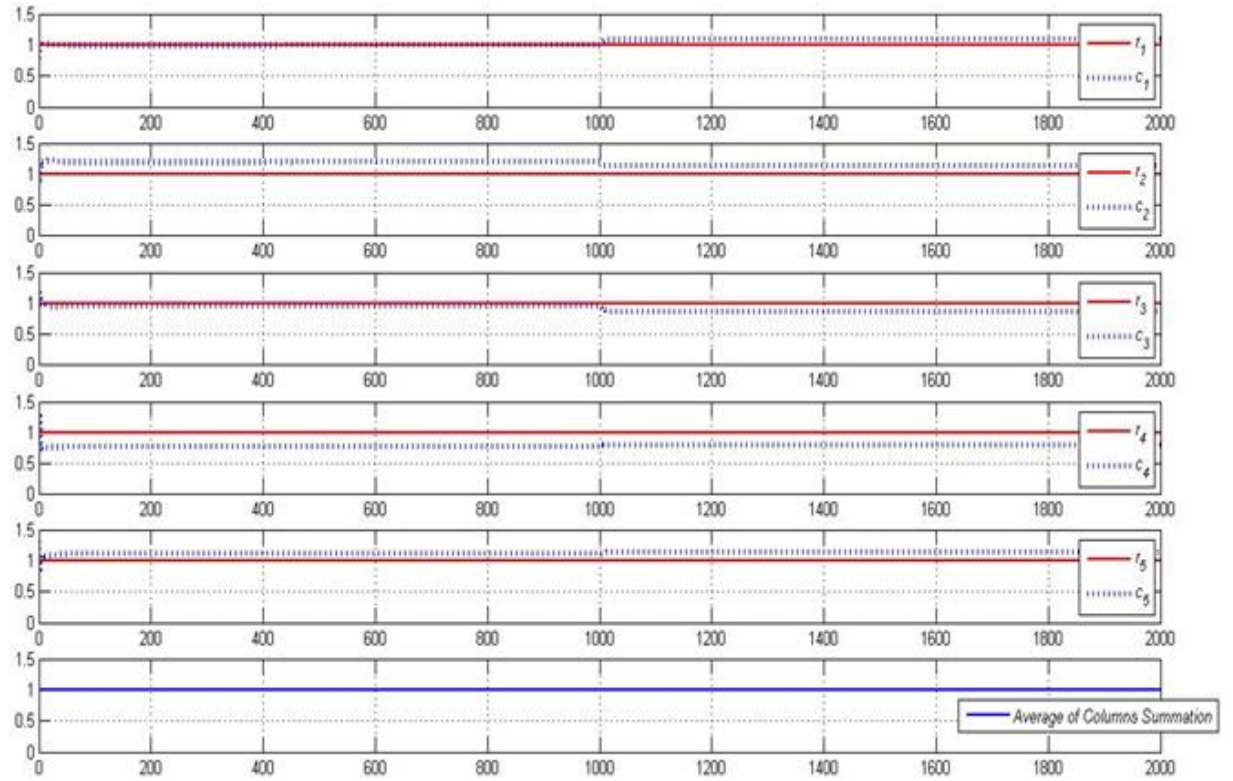


Figure 4.19: Five generators; adjacency rows and columns' summation ($P_o = 10$)

It can be seen this necessity was accomplished when $P_o = 0.01$ in Figure 4.17, but it did not hold as P_o increased as in Figure 4.19. It is noticeable the system reached the consensus even though the sum of columns of the row-stochastic matrix D did not equal one. From this ironic result, it can be concluded that this proposed necessity has not to be held true in order for the system to reach the consensus. Instead, the average of the column's summation of the row-stochastic matrix D must be equal to one.

4.3 IEEE 16-Bus Test Power System for the Load Restoration Problem

In this case, the system contains 16 buses compound of generation and load agents. The system configuration is show in Figure 4.20 below:

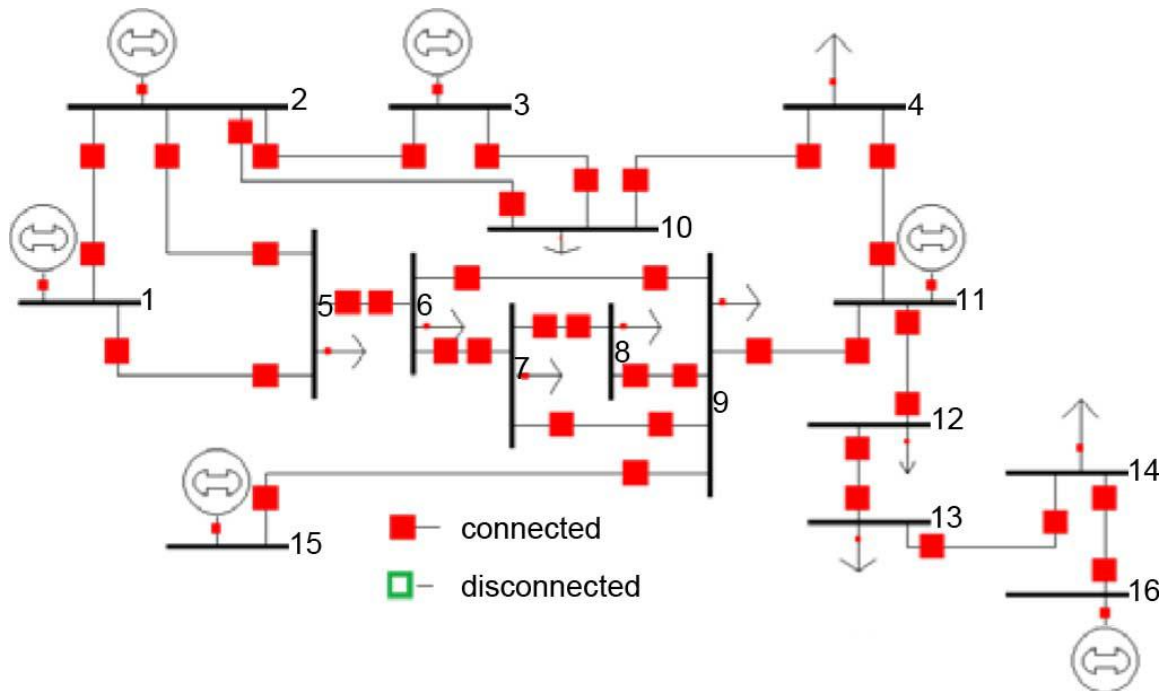


Figure 4.20: The 16-bus test power system (source ref. [18])

It is a standard IEEE test power system. The arrows resemble the power loads, the circles resemble the power generators, the red squares resemble the circuit breakers in the close position, and the green squares resemble the circuit breakers in the open position. In this case of study, the power loses of the transmission lines are neglected.

Data of the 16-bus system in Figure 4.20 on the previous page can be easily summarized in the table 4.3 below [18]:

Table 4.3: 16-Bus power system; system data

Agent or Bus	Neighbors	P_{Gi} [MW]	P_{Li} [MW]	X_i^o [MW]	Priority
1 th	2,5	40	0	40	-
2 th	1,3,5,10	40	0	40	-
3 th	2,10	50	0	50	-
4 th	10,11	0	25	-25	P2
5 th	1,2,6	0	35	-35	P2
6 th	5,7,9	0	20	-20	P1
7 th	6,8,9	0	15	-15	P2
8 th	7,9	0	25	-25	P1
9 th	6,7,8,11,15	0	5	-5	P1
10 th	2,3,4	0	30	-30	P2
11 th	4,9,12	40	0	40	-
12 th	11,13	0	45	-45	P2
13 th	12,14	0	10	-10	P2
14 th	13,16	0	40	-40	P2
15 th	9	30	0	30	-
16 th	14	50	0	50	-

In the table above; P_{Gi} is the agent i^{th} local generation, P_{Li} is the agent i^{th} local load, X_i^o is the agent i^{th} local net power, and $P_{1\sim 2}$ is the agent i^{th} load priority with $P_1 > P_2$. The assumption here according to [20] is a sudden fault occurs to the generator on the bus 11 (agent 11) which led the protection system to react immediately to isolate the fault by opening some circuit breakers. The system configuration of the post fault period is illustrated in Figure 4.21 on the next page.

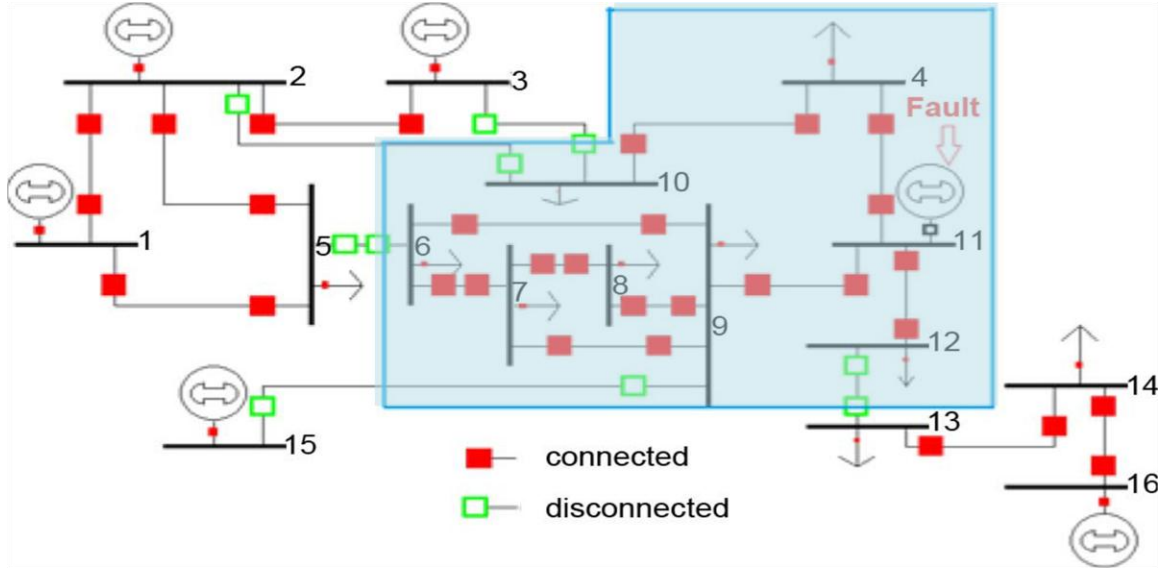


Figure 4.21: The 16-bus system; post fault configuration (source ref. [18])

As a result the power protection reaction, the system lost eight un-faulted loads (out-of-service) which attached to the agents 4 and 6 ~ 12 (shaded area). No doubt, these loads should be restored, as fast as possible, based on the total net power available and their priority. So, the load restoration process has to be activated to return back the lost loads. For such case, very important information like the total net power, the agents' indexes, and the demanded power of the un-faulted loads are extremely required.

To solve the load restoration problem for the situation above in a distribution fashion, the information discovery matrix M is very important. Each agent has to converge to the same M matrix. According to the reference [18], every agent in the system is initially created with the $n \times 3$ matrix. In this initial matrix only two nonzero elements can be found. M_{i1} indicates the agent index (i^{th}) if it is working normally or (0) if it is not. M_{i2} indicates the out-of-service agent index (i^{th}) if it is ready for restoration or (0) if it is not. M_{i3} indicates the local net power for the working agent or the demanded power of the ready for restoration agent. Clearly in the $M_{i,1 \sim 3}$ matrix, the i^{th} agent row could be $[i \ 0 \ P_{Li}]$ if it is working agent or $[0 \ i \ P_{Li}]$ if it is ready for restoration agent.

Because every agent has to converge to the same information matrix M, the information discovery algorithm should be applied to each element of the initial matrices. Here, instead of apply information discovery algorithm based on the calculation methods, which are proposed in the reference [18], a new advanced information discovery algorithm which is based on the estimation methods like, the gradient algorithm, is applied. The final converged matrix M contains all necessary information required for the load restoration. The first column shows all normally working agents, the second column shows all ready for restoration agents, and finally, the third column shows the local net power for the agents whether they are working normally or ready for the restoration as it is shown in the table 4.4 below.

Table 4.4: 16-Bus power system; final converged of information matrix-M

Initials matrices for the agents	Final converged matrix M		
	1 st column	2 nd column	3 rd column
$\begin{bmatrix} 1 & 0 & 40 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 40 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 3 & 0 & 50 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 4 & -25 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 5 & 0 & -35 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 6 & -20 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 7 & -15 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 8 & -25 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 9 & -5 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 10 & -30 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 12 & -45 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 13 & 0 & -10 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 14 & 0 & -40 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 15 & 0 & 30 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 16 & 0 & 50 \end{bmatrix}$	1/16	0	40/16
	2/16	0	40/16
	3/16	0	50/16
	0	4/16	-25/16
	5/16	0	-35/16
	0	6/16	-20/16
	0	7/16	-15/16
	0	8/16	-25/16
	0	9/16	-5/16
	0	10/16	-30/16
	0	0	0/16
	0	12/16	-45/16
	13/16	0	-10/16
	14/16	0	-40/16
	15/16	0	30/16
	16/16	0	50/16

Using the final converged matrix M as input to the 0-1 knapsack problem, the exact ready for the restoration agents with their associated power can be easily identified to reconnect them back to the system as fast as possible. At the end, the balanced system is achieved. The final arrangement of our example is illustrated in Figure 4.22 below for the 0-1knapsack greedy by value. The agents with red circuits were not selected by the 0-1knapsack greedy by value for the restoration.

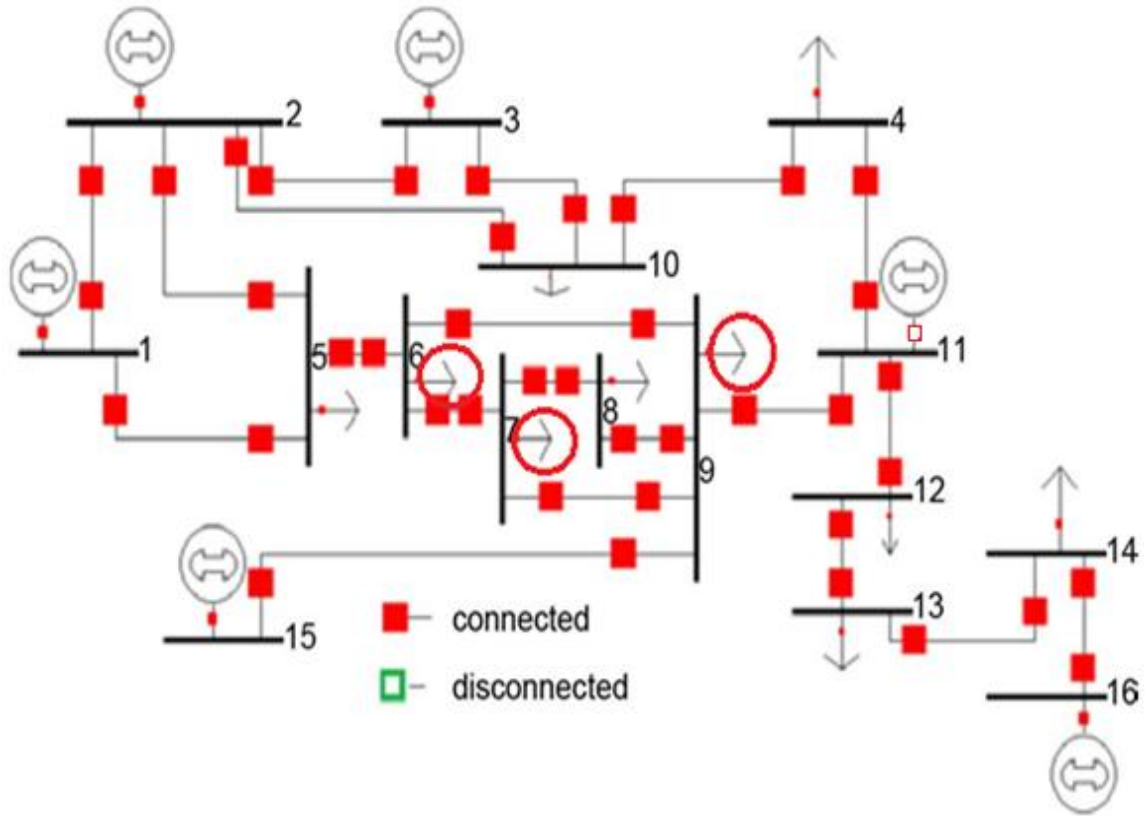


Figure 4.22: The 16-bus system; after restore configuration (source ref. [18])

The operation sequence in the Matlab code of this case was as follows:

- 1- Total Number of iterations $N = 800$.
- 2- Fault occurred at $N = 300$
- 3- Load restored at $N = 500$
- 4- The convergence gradient gain coefficient $\mu = 0.0001$

4.3.1 IEEE 16-Bus Test Power System with New Estimation Method using GA and 0-1 knapsack Value Based

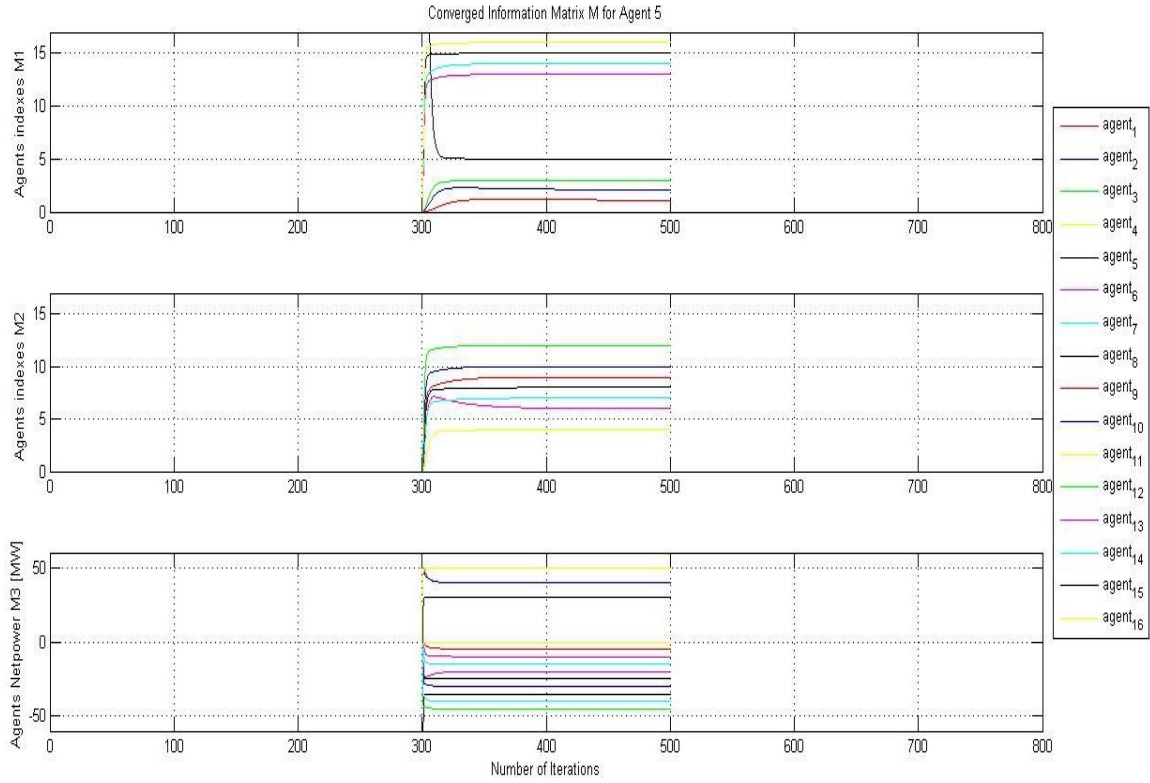


Figure 4.23: The 16-bus system; information matrix-M convergence

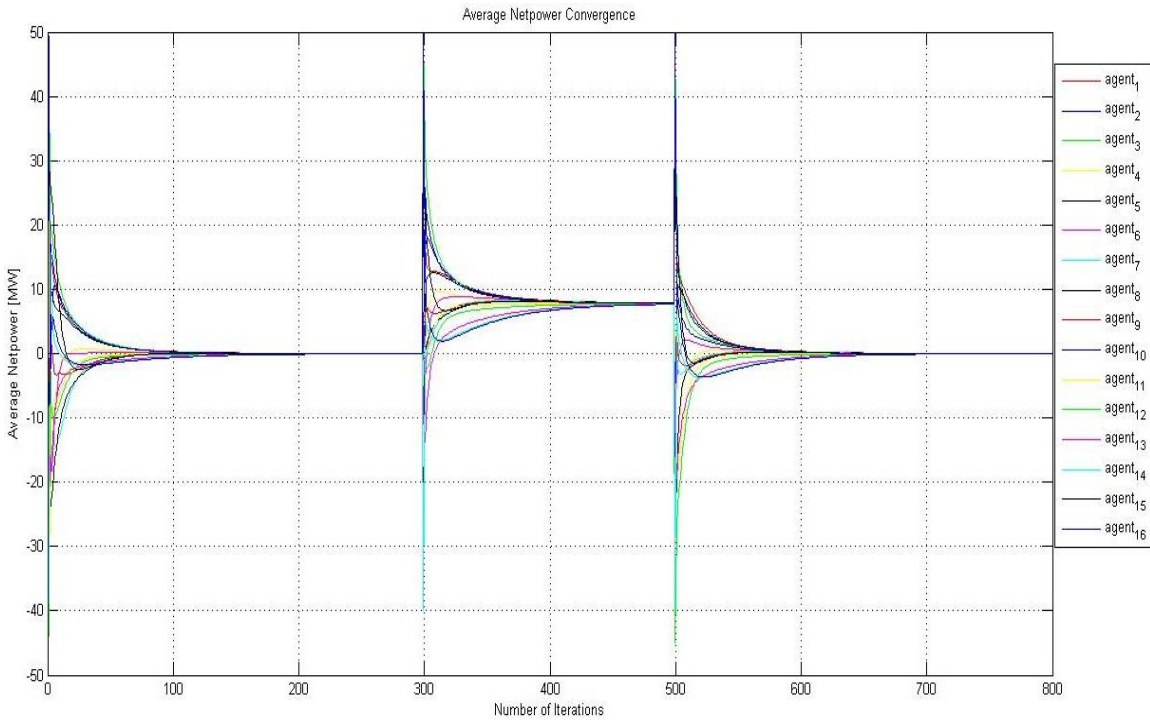


Figure 4.24: The 16-bus system; total net power convergence (knapsack value based)

Instead of using the consensus algorithm based on the calculation methods that were proposed in the reference [18], a new advanced estimation method using the gradient algorithm was applied, and the 16-bus system reached to the same consensus results as that presented in the reference [18]. For instant, the information discovery matrix-M was initiated with initial values corresponding to every agent as in the table 4.4, then it converged to the final converged matrix-M shown in the same table. Figure 4.23 shows the agent convergence process for each matrix column and Figure 4.24 shows the system total average net power convergence in the three stages process; the pre fault, the post fault, and the after fault period. In each stage, the system converged to the net power initial values. When the system was working normally (1~300 iterations in Figure 4.24) all agents converged to zero which is clear if we take the average of the agents in the column five of the table 4.3 on the page 46. During the post fault period (300~500 iterations in Figure 4.24) and because of the out-of-service agents, remains working agents converged to a positive value which is 7.81 MW. Those working agents can be easily identified using the first column of the final converged matrix-M $((40+40+50-35-10-40+30+50)/16 = 7.81)$. Since the average total net power is positive, there is a necessity to restore some, or all out-of-service agents, depending on the system capacity, and their priorities.

Finding such agents (their indexes) for the load restoration is very easy by extracting the information from the second column of the final converged discovery matrix-M. Sometimes and because of power demands' shortage, not all out-of-service agents can be restored. To identify agents that can be restored, the 0-1 knapsack is applied to perform such selection based on the agents' priorities and capacities. After

reconnection out-of-service agents as it is shown in Figure 4.24 (500~800 iterations), the total net power converged back to zero because the 0-1 knapsack value based restored 100% of the demanded power available ($7.81 \times 16 = 125$ MW). The 0-1 knapsack selected agents was; agent 4(-25), agent 8(-25), agent 10(-30), and agent 12(-45).

4.3.2 IEEE 16-Bus Test Power System with New Estimation Method using GA and 0-1 knapsack Density Based

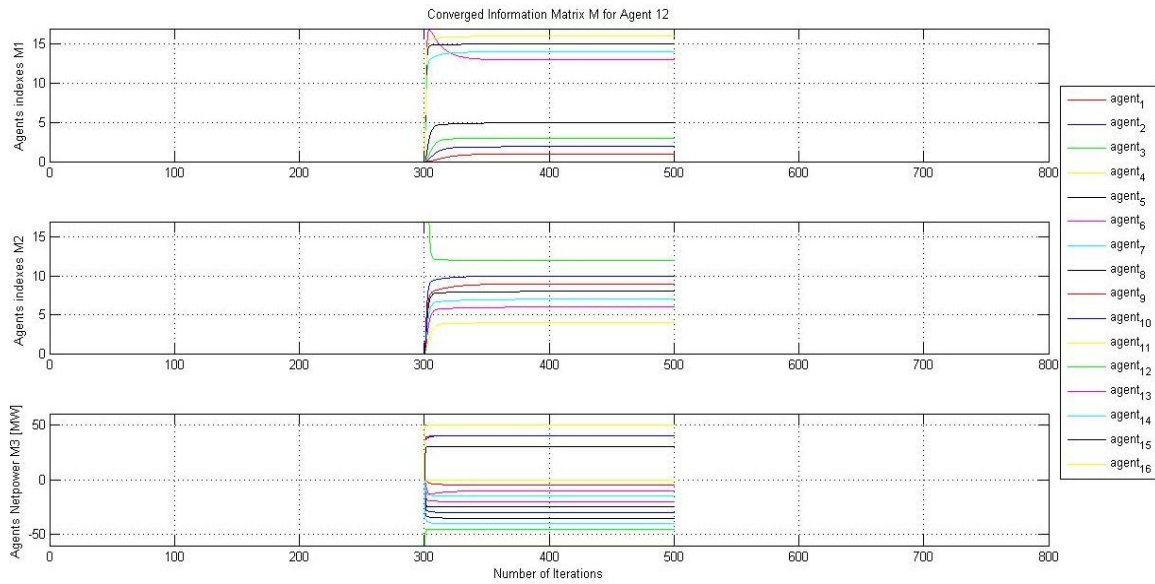


Figure 4.25: The 16-bus system; information matrix-M convergence

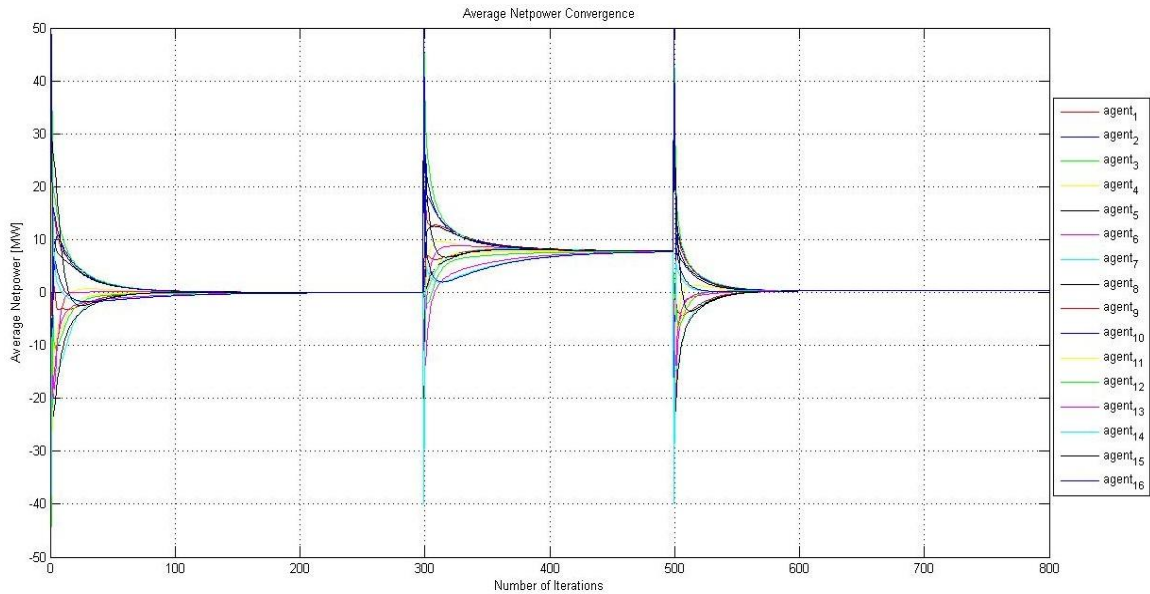


Figure 4.26: The 16-bus system; total net power convergence (knapsack density based)

This time, the 0-1 knapsack density based was applied for the load restoration. By applying the 0-1 knapsack density based, only 96% of the demand power available ($7.81 \times 16 \times 96\% = 124$ MW) was restored. The 0-1 knapsack selected agents was; agent 4(-25), agent 6(-20), agent 7(-15), agent 8(-25), agent 9(-5), and agent 10(-30). As it is seen in Figure 4.27 (500~800 iterations), the average total net power converged to a small positive number which is ($5/16=0.3$) as a result of 96% of the 0-1 knapsack density based.

Both sections 4.3.1 and 4.3.2 prove the capability of using our new advanced consensus algorithm based on the estimation methods like the gradient algorithm for solving such power problem. Moreover, they prove that it is needless to apply traditional calculation methods like those which are presented in the references [5] and [18].

5. Conclusion and Future Work

5.1 Conclusion

In this thesis, a new concept of solving the economic dispatch problem and the load restoration problems is presented. As a part of solving both problems in a distribution fashion, an important information matrix is required for updating system status so a system can reach what is called the consensus. This matrix is called the information discovery matrix, and it is computed based on specific calculation methods.

Instead of using the existing calculation methods of computing the information discovery matrix, adaptive techniques, like the gradient algorithm and the recursive least square algorithm, are applied to estimate above mentioned matrix. This advanced work is extremely useful to save time consumed during the calculation process of reaching consensus especially in the case of the bulk network systems. Our results of reaching the consensus totally matched previous results of cases that had used calculation methods of reaching the consensus. No doubt, this work proves the importance of the estimation methods in the reaching consensus and their capability of solving power problems in a distribution manner which helps to overcome the drawbacks of the central controller.

In this thesis, chapter two gives an important background about the graph and consensus theory. The chapter three gives the theoretical analysis of applying the consensus algorithm to solve the economic dispatch and the load restoration problems. The estimation mathematical model is presented in this chapter. Our case studies and their results are presented in the chapter four. The comments and notes on these results are presented here also. Finally, the chapter five gives the conclusion and the future work of this research.

5.2 Future Work

Some important steps have to be taken as further investigation to complement this work of study in the consensus algorithm based on the estimation techniques. These future researches can be summarized in the following points:

- Study the case of a leader selection in a case if the system is a weighted, an ordered, and a directed graph.
- Study the case of taking into account the power losses in the transmission lines of the power network.
- Study deeply the case of the disconnected agents whether the disconnection happen in the communication network only, the power lines only, or both of them.
- Study the case of losing the system's leader. Dose the system able to select another leader or not? And how?
- Study the case of clastericity when the system is divided to several connected groups and each group converges to its own consensus.

All in all, all these mentioned future researches need more of an advanced code that can take into account many possibilities and assumptions in order get a generally flexible algorithm. Such an algorithm can be easily applied in the real power networks.

REFERENCES

- [1] R. Diestel, *Graph theory*, 3rd ed. Berlin: Springer, 2005.
- [2] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, vol. 95, pp. 215-233, 2007.
- [3] R. Wei, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1859-1864 vol. 3.
- [4] R. A. Gupta and C. Mo-Yuen, "Networked Control System: Overview and Research Trends," *Industrial Electronics, IEEE Transactions on*, vol. 57, pp. 2527-2535, 2010.
- [5] Z. Ziang and C. Mo-Yuen, "Incremental cost consensus algorithm in a smart grid environment," in *Power and Energy Society General Meeting, 2011 IEEE*, 2011, pp. 1-6.
- [6] T. L. Baldwin and E. B. Makram, "Economic wdispatch of electric power systems with line losses," in *System Theory, 1989. Proceedings., Twenty-First Southeastern Symposium on*, 1989, pp. 13-17.
- [7] A. J. Wood and B. F. Wollenberg, *Power generation, operation, and control*, 2nd ed. New York: J. Wiley & Sons, 1996.
- [8] Z. Ziang and C. Mo-Yuen, "Convergence Analysis of the Incremental Cost Consensus Algorithm Under Different Communication Network Topologies in a Smart Grid," *Power Systems, IEEE Transactions on*, vol. 27, pp. 1761-1768, 2012.
- [9] R. J. Wilson, *Introduction to graph theory*, 4th ed. Harlow: Longman, 1996.
- [10] J. A. Bondy, U. S. R. Murty, and SpringerLink (Online service). (2008). *Graph Theory*. Available: <http://0-dx.doi.org.skyline.ucdenver.edu/10.1007/978-1-84628-970-5>
- [11] D. B. West, *Introduction to graph theory*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.
- [12] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *Automatic Control, IEEE Transactions on*, vol. 51, pp. 401-420, 2006.

- [13] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *Automatic Control, IEEE Transactions on*, vol. 51, pp. 1289-1298, 2006.
- [14] R. Olfati-Saber and J. S. Shamma, "Consensus Filters for Sensor Networks and Distributed Sensor Fusion," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 6698-6703.
- [15] V. M. Preciado and G. C. Verghese, "Synchronization in Generalized Erdős-Rényi Networks of Nonlinear Oscillators," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 2005, pp. 4628-4633.
- [16] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *Automatic Control, IEEE Transactions on*, vol. 49, pp. 1465-1476, 2004.
- [17] Z. Ziang and C. Mo-Yuen, "The leader election criterion for decentralized economic dispatch using incremental cost consensus algorithm," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, 2011, pp. 2730-2735.
- [18] X. Yinliang and L. Wenxin, "Novel Multiagent Based Load Restoration Algorithm for Microgrids," *Smart Grid, IEEE Transactions on*, vol. 2, pp. 152-161, 2011.
- [19] J. M. Solanki, S. Khushalani, and N. N. Schulz, "A Multi-Agent Solution to Distribution Systems Restoration," *Power Systems, IEEE Transactions on*, vol. 22, pp. 1026-1034, 2007.
- [20] T. Nagata, Y. Tao, and H. Fujita, "An autonomous agent for power system restoration," in *Power Engineering Society General Meeting, 2004. IEEE*, 2004, pp. 1069-1074 Vol.1.
- [21] Z. Jiangfei, H. TingLei, P. Fei, and L. Yuanjie, "Genetic Algorithm Based on Greedy Strategy in the 0-1 Knapsack Problem," in *Genetic and Evolutionary Computing, 2009. WGECC '09. 3rd International Conference on*, 2009, pp. 105-107.
- [22] M. Radenkovic, "Adaptive Control System Design," class notes for ELEC 5466. Department of Electrical Engineering, University of Colorado Denver, Fall, 2011.

APPENDIX A

MATHEMATICAL DERIVATION OF THE ESTIMATION ALGORITHMS (RLS and GA)

Let us consider the following linear system model [22]:

$$y(i+1) + a_1 y(i) + \dots + a_n y(i+1-n) = b_1 u(i) + b_2 u(i-1) + \dots + b_m u(i+1-m), i = 0, 1, 2, \dots \quad (\text{A.1})$$

Where:

$u(i)$: measurable system input signal

$y(i)$: measurable system output signal

Knowing that q^{-1} is unit delay operator, i.e. $q^{-1}y(i) = y(i-1)$ then system in equation (A.1) becomes:

$$(1 + a_1 q^{-1} + \dots + a_n q^{-n})y(i+1) = (b_1 + b_2 q^{-1} + \dots + b_m q^{-m+1})u(i) \quad (\text{A.2})$$

$$y(i+1) = \frac{B(q^{-1})}{A(q^{-1})} u(i) \quad (\text{A.3})$$

So that:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \dots + a_n q^{-n} \quad (\text{A.4})$$

$$B(q^{-1}) = b_1 + b_2 q^{-1} + \dots + b_m q^{-m+1} \quad (\text{A.5})$$

The term $\frac{B(q^{-1})}{A(q^{-1})}$ in equation (A.3) is transfer operator of the system.

Let us define system parameters and signal vectors as follows:

$$\theta_o^T = [-a_1, -a_2, \dots, -a_n; u(i), \dots, b_m] \quad (\text{A.6})$$

$$\phi(i)^T = [y(i), \dots, y(i+1-n); b_1, \dots, u(i+1-m)] \quad (\text{A.7})$$

So the system is equation (A.1) can be written as:

$$y(i+1) = \phi(i)^T \theta_o \quad (\text{A.8})$$

More general system model is:

$$y(i + 1) = \phi(i)^T \theta_o + d(i) \quad (\text{A.9})$$

Where $d(i)$ represents the cumulative effects of disturbance, measurement errors, etc.

Assuming that system parameters are unknown, an estimate of θ_o is denoted by $\hat{\theta}(i + 1)$ then the following identification model can be constructed:

$$\hat{y}(i + 1) = \hat{\theta}(i + 1)^T \phi(i) \quad (\text{A.10})$$

Estimating $\hat{\theta}(i + 1)$ is determined so that in a certain sense model output $\hat{y}(i + 1)$ matches the system output $y(i + 1)$.

Let us choose $\hat{\theta}(i + 1)$ so that it minimizes the sum of the square errors of the fit.

$$V(\hat{\theta}) = \sum_{k=0}^i \left(y(k + 1) - \phi(k)^T \hat{\theta}(i + 1) \right)^2 \quad (\text{A.11})$$

A necessary condition for a local minimum is:

$$\frac{\partial V(\hat{\theta})}{\partial \hat{\theta}(i+1)} = 0 \text{ which gives:}$$

$$2 \sum_{k=0}^i \left(y(k + 1) - \phi(k)^T \hat{\theta}(i + 1) \right) (-\phi(k)) = 0 \quad (\text{A.12})$$

It follows that:

$$\left(\sum_{k=0}^i \phi(k) \phi(k)^T \right) \hat{\theta}(i + 1) = \sum_{k=0}^i \phi(k) y(k + 1) \quad (\text{A.13})$$

Hence a least square estimate of θ_o is any estimate $\hat{\theta}(i + 1)$ that satisfies equation (A.13).

There may be more than one minimize of $V(\hat{\theta})$.

Let define a new matrix:

$$R(i) \triangleq \sum_{k=0}^i \phi(k) \phi(k)^T \quad (\text{A.14})$$

Which is invertible, and called covariance matrix. then there is an unique least square estimate:

$$\hat{\theta}(i + 1) = R(i)^{-1} \sum_{k=0}^i \phi(k) y(k + 1) \quad (\text{A.15})$$

A.1 Derivation of Recursive Least Square Algorithm Estimation (RLS)

Considering the least square estimate in equation (A.13), we wish to determine a recursive form of this estimate that suitable for easily updating $\hat{\theta}(i)$ to $\hat{\theta}(i + 1)$.

Recalling equation (A.14), we see that

$$R(i) = R(i - 1) + \phi(i)\phi(i)^T \quad (\text{A.16})$$

Then from (A.13)

$$R(i)\hat{\theta}(i + 1) = \sum_{k=0}^i \phi(k) y(k + 1) \quad (\text{A.17})$$

$$R(i)\hat{\theta}(i + 1) = \sum_{k=0}^{i-1} \phi(k) y(k + 1) + \phi(i)y(i + 1) \quad (\text{A.18})$$

$$R(i)\hat{\theta}(i + 1) = R(i - 1)\hat{\theta}(i) + \phi(i)y(i + 1) \text{ from (A.13)} \quad (\text{A.19})$$

$$R(i)\hat{\theta}(i + 1) = (R(i) - \phi(i)\phi(i)^T) \hat{\theta}(i) + \phi(i)y(i + 1) \text{ from (A.16)} \quad (\text{A.20})$$

Thus:

$$R(i)\hat{\theta}(i + 1) = R(i)\hat{\theta}(i) - \phi(i)\phi(i)^T \hat{\theta}(i) + \phi(i)y(i + 1) \quad (\text{A.21})$$

$$R(i)\hat{\theta}(i + 1) = R(i)\hat{\theta}(i) + \phi(i)[y(i + 1) - \phi(i)^T \hat{\theta}(i)] \quad (\text{A.22})$$

Then:

$$\hat{\theta}(i + 1) = \hat{\theta}(i) + R(i)^{-1}\phi(i)[y(i + 1) - \phi(i)^T \hat{\theta}(i)] \quad (\text{A.23})$$

From stored values of $\hat{\theta}(i)$ and $R(i - 1)$, and new observations $y(i + 1)$ and $\theta(i)$, by using equations (A.20 & A.23) we can calculate the new values $\hat{\theta}(i + 1)$ and $R(i)$.

Equations (A.16 & A.23) are called the Recursive Least Square Estimate (RLS).

To avoid inverting matrix $R(i)^{-1}$ in equation (A.23), we use what called the matrix invariance lemma:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \quad (\text{A.25})$$

Here, all matrices inverses on the right-hand side are exist.

Let define $P(i) \triangleq R(i)^{-1}$ then from (A.17)

$$P(i)^{-1} = P(i-1)^{-1} + \phi(i)\phi(i)^T \quad (\text{A.26})$$

By applying the rule of matrix invariance lemma, we set:

$A = P(i)^{-1}$, $B = \phi(k)$, $C = 1$, and $D = \phi(k)^T$ then the equation (A.25) becomes

$$P(i) = P(i-1) + \frac{P(i-1)\phi(k)\phi(i)^T P(i-1)}{1 + \phi(i)^T P(i-1)\phi(k)} \quad (\text{A.27})$$

Then we have obtained the following RLS estimate that requires no inversion to be performed at each step.

$$\hat{\theta}(i+1) = \hat{\theta}(i) + P(i)\phi(i)[y(i+1) - \phi(i)^T \hat{\theta}(i)] \quad (\text{A.28})$$

$$P(i) = P(i-1) + \frac{P(i-1)\phi(k)\phi(i)^T P(i-1)}{1 + \phi(i)^T P(i-1)\phi(k)} \quad (\text{A.29})$$

$$P(0) = P(0)^T > 0$$

A.2 Derivation of Gradient Algorithm Estimation (GA)

Given an estimate $\hat{\theta}$, the squared prediction error is:

$$e(i)^2 = \frac{1}{2} (y(i) - \phi(i-1)^T \hat{\theta})^2 \quad (\text{A.30})$$

Its gradient with respect to $\hat{\theta}$ is:

$$\nabla_{\theta} e(i)^2 = -\phi(i-1)[y(i) - \phi(i-1)^T \hat{\theta}] \quad (\text{A.31})$$

The direction of the negative gradient $\phi(i-1)[y(i) - \phi(i-1)^T \hat{\theta}]$ is the direction in which a small change in $\hat{\theta}$ will lead to a decrease in $e(i)^2$. Thus one motivated to consider the algorithm

$$\hat{\theta}(i) = \hat{\theta}(i-1) + \mu \phi(i-1) (y(i) - \phi(i-1)^T \hat{\theta}(i-1)) \quad (\text{A.32})$$

Where $\mu > 0$ is a gain to increase the speed of the convergence.

Equation (A.32) is called the Gradient Algorithm Estimate (GA) [22].

A.3 Basic Flowchart for the Estimation Algorithms (RLS and GA)

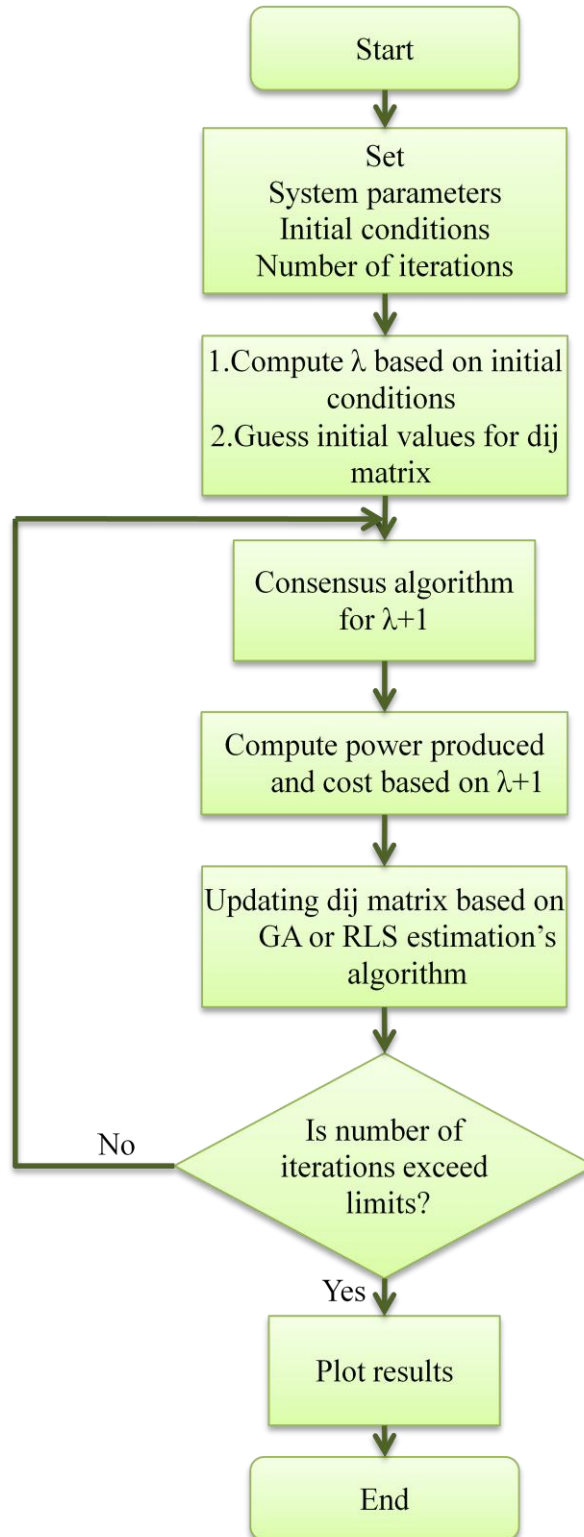


Figure A.1: Flowchart of solving EDP based on new estimation algorithms